Basic arithmetic in Flint and Nemo

William Hart, Fredrik Johansson
Tommy Hofmann, Claus Fieker

October 23, 2017

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Univariate polynomials over the above rings

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Univariate polynomials over the above rings
- Linear algebra over the above rings

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Univariate polynomials over the above rings
- Linear algebra over the above rings
- Primality testing and integer factorisation

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Univariate polynomials over the above rings
- Linear algebra over the above rings
- Primality testing and integer factorisation
- Polynomial factorisation over $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Univariate polynomials over the above rings
- Linear algebra over the above rings
- Primality testing and integer factorisation
- Polynomial factorisation over $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$
- LLL, HNF, SNF

Introduction to Flint

- Basic rings: $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$, $\mathbb{Q}_p$
- Univariate polynomials over the above rings
- Linear algebra over the above rings
- Primality testing and integer factorisation
- Polynomial factorisation over $\mathbb{Z}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathbb{F}_q$
- LLL, HNF, SNF
- (Work in progress) multivariate polynomials

New features in Flint

- Quadratic sieve integer factorisation

New features in Flint

- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation

New features in Flint

- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation
- APRCL primality test

New features in Flint

- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation
- APRCL primality test
- Parallelised FFT

New features in Flint

- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation
- APRCL primality test
- Parallelised FFT
- Howell form

New features in Flint

- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation
- APRCL primality test
- Parallelised FFT
- Howell form
- Characteristic and minimal polynomial

New features in Flint

- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation
- APRCL primality test
- Parallelised FFT
- Howell form
- Characteristic and minimal polynomial
- van Hoeij factorisation for $\mathbb{Z}[x]$

New features in Flint

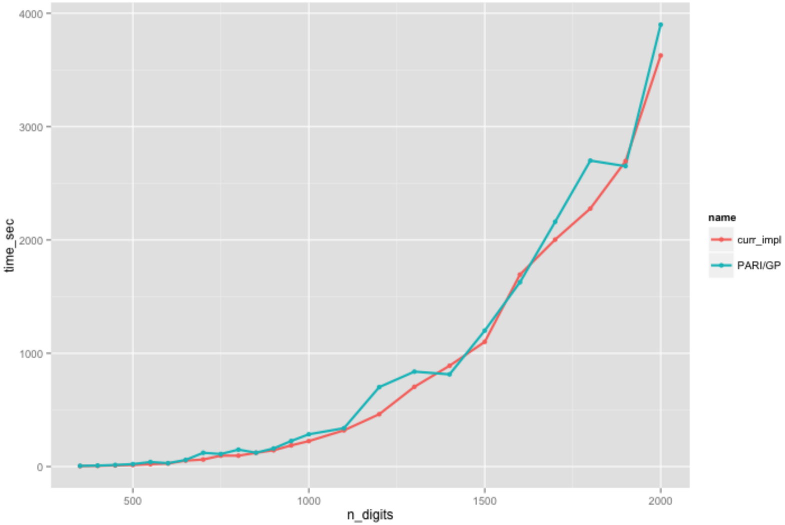
- Quadratic sieve integer factorisation
- Elliptic curve integer factorisation
- APRCL primality test
- Parallelised FFT
- Howell form
- Characteristic and minimal polynomial
- van Hoeij factorisation for $\mathbb{Z}[x]$
- Multivariate polynomial arithmetic $\mathbb{Z}[x, y, z, \ldots]$

Integer factorisation : Quadratic sieve

Table : Quadratic sieve timings

Digits	Pari/GP	Flint (1 core)	Flint (4 cores)
50	0.43	0.55	0.39
59	3.8	3.0	1.7
68	38	21	14
77	257	140	52
83	2200	1500	540

APRCL primality test timings



FFT: Integer and polynomial multiplication

Table : FFT timings

Words	1 core	4 cores	8 cores
110k	0.07s	0.05s	0.05s
360k	0.3s	0.1	0.1s
1.3m	1.1s	0.4s	0.3s
4.6m	4.5s	1.5s	1.0s
26m	28s	9s	6s
120m	140s	48s	33s
500m	800s	240s	150s

Characteristic and minimal polynomial

Table : Charpoly and minpoly timings

Op	Sage 6.9	Pari 2.7.4	Magma 2.21-4	Giac 1.2.2	Flint
Charpoly	0.2s	0.6s	0.06s	0.06s	0.04s
Minpoly	0.07s	>160 hrs	0.05s	0.06s	0.04s

for 80×80 matrix over $\mathbb{Z}$ with entries in $[-20, 20]$ and minpoly of degree 40.

Multivariate polynomials

- Joint work with Daniel Schultz

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...
- Sparse: heap based algorithms (Monagan and Pearce)

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...
- Sparse: heap based algorithms (Monagan and Pearce)
- "Dense": big array method

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...
- Sparse: heap based algorithms (Monagan and Pearce)
- "Dense": big array method
- Parallel multiplication - diagonal splitting (Gastineau)

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...
- Sparse: heap based algorithms (Monagan and Pearce)
- "Dense": big array method
- Parallel multiplication - diagonal splitting (Gastineau)
- Fast assembly for accumulation into 3 limbs

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...
- Sparse: heap based algorithms (Monagan and Pearce)
- "Dense": big array method
- Parallel multiplication - diagonal splitting (Gastineau)
- Fast assembly for accumulation into 3 limbs
- Pack monomials using Kronecker segmentation

Multivariate polynomials

- Joint work with Daniel Schultz
- Only over $\mathbb{Z}$ so far...
- Sparse: heap based algorithms (Monagan and Pearce)
- “Dense”: big array method
- Parallel multiplication - diagonal splitting (Gastineau)
- Fast assembly for accumulation into 3 limbs
- Pack monomials using Kronecker segmentation
- Support lex, deglex and degrevlex, exponents up to 63 bits

Multivariate multiplication

Table : "Dense" Fateman multiply bench

n	Trip (POLH)	Flint
4	0.13ms	0.11ms
6	0.29ms	0.45ms
8	0.91ms	1.5ms
10	3.2ms	4.4ms
12	10ms	10ms

4 variables

Multivariate multiplication

Table : "Dense" Fateman multiply bench

n	Sage	Singular	Magma	Giac	Piranha	Trip	Flint
5	0.0063s	0.0048s	0.0018s	0.00023s	0.0011s	0.00057s	0.00023s
10	0.51s	0.11s	0.12s	0.0056s	0.029s	0.023s	0.0043s
15	9.1s	1.4s	1.9s	0.11s	0.39s	0.21s	0.045s
20	75s	21s	16s	0.62s	2.9s	2.3s	0.48s
25	474s	156s	98s	2.8s	14s	12s	2.3s
30	1667s	561s	440s	14s	56s	41s	10s

4 variables

Multivariate multiplication

Table : Sparse multiply benchmark

n	Sage	Singular	Magma	Giac	Piranha	Trip	Flint
4	0.0066s	0.0050s	0.0062s	0.0046s	0.0033s	0.0015s	0.0014s
6	0.15s	0.11s	0.080s	0.030s	0.025s	0.016s	0.016s
8	1.6s	0.79s	0.68s	0.28s	0.15s	0.10s	0.10s
10	8s	3.6s	3.0s	1.5s	0.62s	0.40s	0.48s
12	43s	14s	11s	4.8s	2.2s	2.2s	2.0s
14	173s	63s	37s	14s	6.7s	12s	7.2s
16	605s	201s	94s	39s	20s	39s	19s

5 variables

Multivariate multiplication

Table : Sparse Pearce 2 core

n	Giac	Piranha	Trip	Flint
4	0.0070s	0.0033s	0.0015s	0.0016s
6	0.044s	0.025s	0.016s	0.012s
8	0.35s	0.11s	0.088s	0.070s
10	1.5s	0.33s	0.33s	0.30s
12	4.8s	1.19s	1.52s	1.16s
14	14s	3.6s	7.5s	3.9s
16	35s	10.7s	21s	11.5s

4 variables

Multivariate multiplication

Table : Sparse Pearce 4 core

n	Giac	Piranha	Trip	Flint
4	0.0062s	0.0034s	0.0015s	0.0013s
6	0.034s	0.025s	0.016s	0.011s
8	0.31s	0.078s	0.093s	0.047s
10	1.2s	0.23s	0.32s	0.19s
12	3.6s	0.71s	1.2s	0.70s
14	10.5s	2.0s	5.5s	2.5s
16	25s	5.7s	10.3s	6.7s

4 variables

Exact quotient

Table : "Dense" quotient only

n	Sage	Singular	Magma	Giac	Flint
5	0.02s	0.003s	0.002s	0.0002s	0.0001s
10	1.1s	0.11s	0.16s	0.0039s	0.0022s
15	28s	1.5s	3.5s	0.049	0.022s
20	340s	19s	35s	0.25s	0.15s
25	2500s	130s	210s	1.1s	0.93s
30	——	470s	830s	6.0s	3.6s

4 variables

Exact quotient

Table : Sparse quotient only

n	Sage	Singular	Magma	Giac	Flint
4	0.46s	0.01s	0.005s	0.001s	0.0008s
6	77s	0.15s	0.17s	0.014s	0.010s
8	——	1.3s	3.1s	0.12s	0.068s
10	——	8.1s	27s	0.93s	0.35s
12	——	37s	140s	2.5s	1.7s
14	——	144s	630s	8.0s	6.6s
16	——	514s	2300s	22s	18s

5 variables

Divisibility testing

Table : "Dense" divisibility test with quotient

n	Sage	Singular	Magma	Giac	Flint
5	0.02s	0.006s	0.002s	0.001s	0.0005s
10	1.1s	0.56s	0.16s	0.05s	0.020s
15	28s	15s	3.3s	0.15s	0.054s
20	340s	150s	31s	0.90s	0.48s
25	2500s	840s	200s	4.1s	2.3s
30	——	3100s	830s	21s	11s

4 variables, returns quotient

Divisibility testing

Table : Sparse divisibility test with quotient

n	Sage	Singular	Magma	Giac	Flint
4	0.49s	0.03s	0.005s	0.002s	0.002s
6	107s	0.54s	0.17s	0.024s	0.024s
8	——	6.6s	3.1s	0.19s	0.16s
10	——	38s	27s	1.3s	0.74s
12	——	160s	140s	4.3s	3.2s
14	——	600s	630s	14s	14s
16	——	1900s	2300s	40s	40s

5 variables, returns quotient

Multivariate multiplication

Table : Sparse Pearce 1 core

n	Maple	Sdmp	Flint
4	0.0010s	0.0010s	0.0010s
6	0.013s	0.012s	0.012s
8	0.080s	0.074s	0.078s
10	0.35s	0.32s	0.34s
12	2.1s	1.2s	1.2s
14	14s	3.6s	3.7s
16	52s	9.6s	10s

4 variables

Multivariate multiplication

Table : Sparse Pearce 2 core

n	Maple	Sdmp	Flint
4	0.0020s	0.0017s	0.00084s
6	0.012s	0.0094s	0.0077s
8	0.065s	0.060s	0.047s
10	0.28s	0.26s	0.20s
12	1.60s	0.93s	0.73s
14	12s	2.7s	2.5s
16	52s	6.8s	6.6s

4 variables

Multivariate multiplication

Table : Sparse Pearce 4 core

n	Maple	Sdmp	Flint
4	0.0020s	0.0017s	0.00066s
6	0.014s	0.010s	0.0049s
8	0.058s	0.056s	0.028s
10	0.23s	0.20s	0.11s
12	1.40s	0.72s	0.45s
14	12s	2.2s	1.7s
16	48s	5.0s	4.4s

4 variables

Introducing

A computer algebra package for the Julia programming language.

http://nemocas.org/

Language for generic programming

- Support for Windows, Linux, Mac

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading
- Fast generics and metaprogramming

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading
- Fast generics and metaprogramming
- Maintained and popular

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading
- Fast generics and metaprogramming
- Maintained and popular
- Open source

Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading
- Fast generics and metaprogramming
- Maintained and popular
- Open source
- Imperative syntax

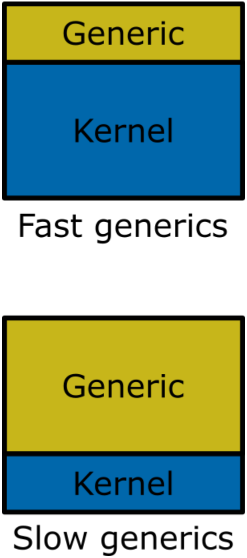
Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading
- Fast generics and metaprogramming
- Maintained and popular
- Open source
- Imperative syntax
- Garbage collected

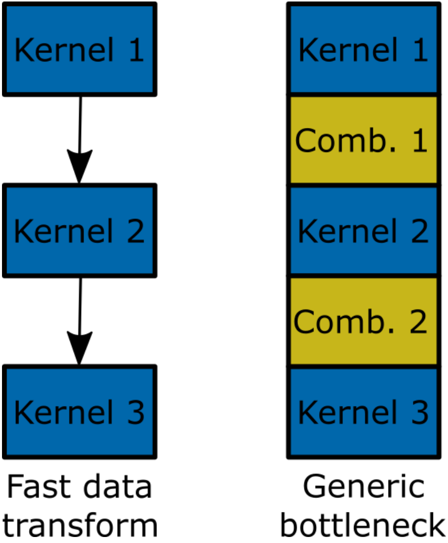
Language for generic programming

- Support for Windows, Linux, Mac
- 64 bit integers and double precision floats
- Console/REPL mode
- Operator overloading
- Fast generics and metaprogramming
- Maintained and popular
- Open source
- Imperative syntax
- Garbage collected
- Easy/efficient C interop

Efficient generics



Efficient generics



julia

- JIT compilation : near C performance.
- Designed by mathematically minded people.
- Open Source (MIT License).
- Actively developed since 2009.
- Supports Windows, OSX, Linux, BSD.
- Friendly C/Python-like (imperative) syntax.

Julia is polymorphic:

```
gcd(a::Int, b::Int)
gcd(a::BigInt, b::BigInt)
gcd{T <: Field}(a::Poly{T}, b::Poly{T})
```

Julia is polymorphic:

```
gcd(a::Int, b::Int)
gcd(a::BigInt, b::BigInt)
gcd{T <: Field}(a::Poly{T}, b::Poly{T})
```

Julia supports multimethods:

```
*(a::Int, b::Matrix{Int})
*(a::Matrix{Int}, b::Int)
```

Julia supports triangular dispatch:

```
*{T <: QuotientRing, S <: Poly{T}}(x::T, y::S)
```

Julia supports coercion in a natural way:

```
+{T <: Domain}(a::Laurent{T}, b::Series{FractionField{T}})
```

Julia supports triangular dispatch:

```
*{T <: QuotientRing, S <: Poly{T}}(x::T, y::S)
```

Julia supports coercion in a natural way:

```
+{T <: Domain}(a::Laurent{T}, b::Series{FractionField{T}})
```

Julia supports:

- Custom array indexing
- Custom printing of objects
- Custom promotion rules and conversions

Julia supports triangular dispatch:

```
*{T <: QuotientRing, S <: Poly{T}}(x::T, y::S)
```

Julia supports coercion in a natural way:

```
+{T <: Domain}(a::Laurent{T}, b::Series{FractionField{T}})
```

Julia supports:

- Custom array indexing
- Custom printing of objects
- Custom promotion rules and conversions

Coming soon in Julia:

- Traits

Interfaces to C libraries:

- Flint : univariate polys and matrices over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/p\mathbb{Z}$, F_q, Q_p

Interfaces to C libraries:

- Flint : univariate polys and matrices over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/p\mathbb{Z}$, F_q, Q_p
- Arb : ball arithmetic, univariate polys and matrices over $\mathbb{R}$ and $\mathbb{C}$, special and transcendental functions

Interfaces to C libraries:

- Flint : univariate polys and matrices over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/p\mathbb{Z}$, F_q, Q_p
- Arb : ball arithmetic, univariate polys and matrices over $\mathbb{R}$ and $\mathbb{C}$, special and transcendental functions
- Antic : element arithmetic over abs. number fields

Interfaces to C libraries:

- Flint : univariate polys and matrices over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/p\mathbb{Z}$, F_q, Q_p
- Arb : ball arithmetic, univariate polys and matrices over $\mathbb{R}$ and $\mathbb{C}$, special and transcendental functions
- Antic : element arithmetic over abs. number fields

Nemo capabilities:

- Generic rings: residue rings, fraction fields, dense univariate polynomials, sparse distributed multivariate polynomials

Interfaces to C libraries:

- Flint : univariate polys and matrices over $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/p\mathbb{Z}$, F_q, Q_p
- Arb : ball arithmetic, univariate polys and matrices over $\mathbb{R}$ and $\mathbb{C}$, special and transcendental functions
- Antic : element arithmetic over abs. number fields

Nemo capabilities:

- Generic rings: residue rings, fraction fields, dense univariate polynomials, sparse distributed multivariate polynomials , dense linear algebra, power series (absolute and relative), permutation groups

Highlights of Nemo

Highlights:

- Generic polynomial resultant (Ducos)

Highlights of Nemo

Highlights:

- Generic polynomial resultant (Ducos)
- charpoly, minpoly over an integrally closed domain

Highlights of Nemo

Highlights:

- Generic polynomial resultant (Ducos)
- charpoly, minpoly over an integrally closed domain
- Smith and Hermite normal form, Popov form

Highlights of Nemo

Highlights:

- Generic polynomial resultant (Ducos)
- charpoly, minpoly over an integrally closed domain
- Smith and Hermite normal form, Popov form
- fast generic determinant

Highlights of Nemo

Highlights:

- Generic polynomial resultant (Ducos)
- charpoly, minpoly over an integrally closed domain
- Smith and Hermite normal form, Popov form
- fast generic determinant
- division/fraction free, interpolation methods for linear algebra

Highlights of Nemo

Highlights:

- Generic polynomial resultant (Ducos)
- charpoly, minpoly over an integrally closed domain
- Smith and Hermite normal form, Popov form
- fast generic determinant
- division/fraction free, interpolation methods for linear algebra
- fast sparse multivariate arithmetic (Monagan and Pearce)

Singular.jl

Access to Singular kernel functions and data types:

- Coefficient rings $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, $\mathrm{GF}(p)$, etc.

Singular.jl

Access to Singular kernel functions and data types:

- Coefficient rings $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, GF(p), etc.
- Polynomials, ideals, modules, matrices, etc.

Singular.jl

Access to Singular kernel functions and data types:

- Coefficient rings $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, GF(p), etc.
- Polynomials, ideals, modules, matrices, etc.
- Groebner basis, resolutions, syzygies

Singular.jl

Access to Singular kernel functions and data types:

- Coefficient rings $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, GF(p), etc.
- Polynomials, ideals, modules, matrices, etc.
- Groebner basis, resolutions, syzygies

Integration with Nemo.jl:

- Singular polynomials over any Nemo coefficient ring, e.g. Groebner bases over cyclotomic fields

Singular.jl

Access to Singular kernel functions and data types:

- Coefficient rings $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{Z}/n\mathbb{Z}$, GF(p), etc.
- Polynomials, ideals, modules, matrices, etc.
- Groebner basis, resolutions, syzygies

Integration with Nemo.jl:

- Singular polynomials over any Nemo coefficient ring, e.g. Groebner bases over cyclotomic fields
- Nemo generics over any Singular ring

Demo

Demo...

Generic benchmarks

Iterated univariate arithmetic measures generic performance

Generic benchmarks

Iterated univariate arithmetic measures generic performance (all systems have optimised univariate arithmetic for $\mathbb{Z}[x]$)

Generic benchmarks

Iterated univariate arithmetic measures generic performance (all systems have optimised univariate arithmetic for $\mathbb{Z}[x]$)

- $f = (t + (z + (y + (x + 1))))$
- $p = f^{30}$
- time $q = p * (p + 1)$

Table : Dense Recursive Fateman $Z[x][y][z][t]$

Sage	Pari/GP	Magma	Nemo
132s	156s	233s	44s

Generic benchmarks

- $f = (5u^5 + (3t^3t + (2z^2 + (y + (x + 1)))))^{16}$
- $g = (u + (t + (2z^2 + (3y^3 + (5x^5 + 1)))))^{16}$
- time $q = f * g$

Table : Pearce $Z[x][y][z][t][u]$

Sage	Pari/GP	Magma	Nemo
2900s	798s	647s	167s

- $R\langle x\rangle = GF(17^{11})$
- $S = R[y]$

- $R\langle x\rangle = GF(17^{11})$
- $S = R[y]$
- $T = S/(y^3 + 3xy + 1)$
- $U = T[z]$

- $R\langle x\rangle = GF(17^{11})$
- $S = R[y]$
- $T = S/(y^3 + 3xy + 1)$
- $U = T[z]$
- $f = T(3y^2 + y + x)z^2 + T((x+2)y^2 + x + 1)z + T(4xy + 3)$
- $g = T(7y^2 - y + 2x + 7)z^2 + T(3y^2 + 4x + 1)z + T((2x+1)y + 1)$
- $s = f^{12}$
- $t = (s + g)^{12}$

- $R\langle x\rangle = GF(17^{11})$
- $S = R[y]$
- $T = S/(y^3 + 3xy + 1)$
- $U = T[z]$
- $f = T(3y^2 + y + x)z^2 + T((x+2)y^2 + x + 1)z + T(4xy + 3)$
- $g = T(7y^2 - y + 2x + 7)z^2 + T(3y^2 + 4x + 1)z + T((2x+1)y + 1)$
- $s = f^{12}$
- $t = (s + g)^{12}$
- time $r =$ resultant(s, t)

Table : Resultant

Sage	Pari/GP	Magma	Nemo
179907s	N/A	82s	0.2s

Generic benchmarks

Benchmark for generic power series

- $R = \mathbb{Q}[x]$
- $S = R[[t]]$
- $u = t + O(t^{1000})$
- time $r = (u \exp(xu))/(\exp(u) - 1)$

Table : Bernoulli polynomials

Sage	Pari/GP	Magma	Nemo
161s	235s	4223s	65s

Generic benchmarks

Generic polynomials over Antic number field elements

- $R\langle x\rangle = \text{CyclotomicField}(20)$
- $S = R[y]$
- $f = (3x^7 + x^4 - 3x + 1)y^3 + (2x^6 - x^5 + 4x^4 - x^3 + x^2 - 1)y + (-3x^7 + 2x^6 - x^5 + 3x^3 - 2x^2 + x)$
- time $r = f^{300}$

Table : Polynomials over a number field

Sage	Pari/GP	Magma	Nemo
6.92s	0.21s	0.70s	0.13s

Generic benchmarks

- $n = 2003 \times 1009$
- $R = (\mathbb{Z}/n\mathbb{Z})[x]$
- $M = (a_{i,j}) \in \mathrm{Mat}_{80\times 80}(R)$, $\deg(a_{i,j}) \leq 5$, $||a_{i,j}||_\infty \leq 100$
- time determinant(M)

Table : Determinant over commutative ring

Sage	Pari/GP	Magma	Nemo
43.5s	456s	est. $> 4 \times 10^{19}$s	7.5s

- $K\langle a\rangle = \mathrm{NumberField}(x^3 + 3x + 1)$
- $M = (a_{i,j}) \in \mathrm{Mat}_{80\times 80}(K)$, $\deg(a_{i,j}) = 2$, $||a_{i,j}||_\infty \leq 100$
- time determinant(M)

There is coefficient blowup in this example.

Table : Determinant over number field

Sage	Pari/GP	Magma	Nemo
5893s	21.9s	5.3s	2.4s

- $R = \mathbb{Z}[x]$
- $M = (a_{i,j}) \in \text{Mat}_{40\times 40}(R)$, $\deg(a_{i,j}) = 2$, $||a_{i,j}||_\infty \leq 20$
- time determinant(M)

There is coefficient blowup in this example.

Table : Determinant over a polynomial ring

Sage	Pari/GP	Magma	Nemo
6.3s	1.3s	3.2s	0.24s

- $R = \mathbb{Z}[x][y]$
- $M = (a_{i,j}) \in \text{Mat}_{20\times 20}(R)$, $\deg(a_{i,j}) = 2, 2$, $||a_{i,j}||_\infty \leq 20$
- $b = (a_1, a_2, \ldots, a_{20})^T$, entries as for M
- time solve $Mx = b$

There is coefficient blowup in this example.

Table : Linear solve over (fraction field of) polynomial ring

Sage	Pari/GP	Magma	Nemo
$> 10^5$s	$> 10^6$s	90s	7s

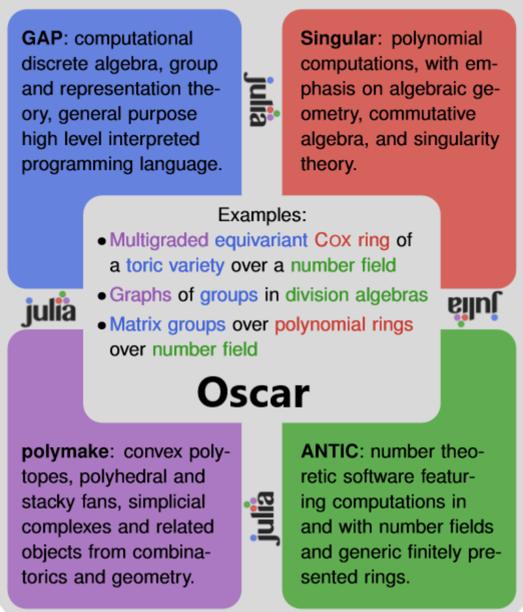
- $R = \mathbb{Z}[x]$
- $M = (a_{i,j}) \in \text{Mat}_{20\times 20}(R)$, block diagonal with two 10×10 blocks, $\deg(a_{i,j}) = 2$, $||a_{i,j}||_\infty \leq 20$
- apply ten "small" random similarity transforms
- time minpoly(M)

Table : Minimal polynomial over integrally closed gcd domain

Sage	Pari/GP	Magma	Nemo
Exception	$> 6 \times 10^6$s	N/A	0.6s

OSCAR (Open Source Computer Algebra Resource)

Develop a visionary, next generation, open source computer algebra system, integrating all systems, libraries and packages developed within the TRR.


GAP: computational discrete algebra, group and representation theory, general purpose high level interpreted programming language.
Singular: polynomial computations, with emphasis on algebraic geometry, commutative algebra, and singularity theory.
Examples:
• Multigraded equivariant Cox ring of a toric variety over a number field
• Graphs of groups in division algebras
• Matrix groups over polynomial rings over number field
Oscar
polymake: convex polytopes, polyhedral and stacky fans, simplicial complexes and related objects from combinatorics and geometry.
ANTIC: number theoretic software featuring computations in and with number fields and generic finitely presented rings.
julia
julia
julia
julia

Antic cornerstone

C libraries:

- Flint - polynomials and linear algebra

Antic cornerstone

C libraries:

- Flint - polynomials and linear algebra
- Antic - number field arith.

Antic cornerstone

C libraries:

- Flint - polynomials and linear algebra
- Antic - number field arith.
- MPIR (fork of GMP) - bignum arithmetic

Antic cornerstone

C libraries:

- Flint - polynomials and linear algebra
- Antic - number field arith.
- MPIR (fork of GMP) - bignum arithmetic

Julia libraries:

- Nemo.jl - generic, finitely presented rings

Antic cornerstone

C libraries:

- Flint - polynomials and linear algebra
- Antic - number field arith.
- MPIR (fork of GMP) - bignum arithmetic

Julia libraries:

- Nemo.jl - generic, finitely presented rings
- Hecke.jl - number fields, class field theory, algebraic number theory

Thank You

http://nemocas.org/