Experiments and expectations in computational algebra

John Voight
University of Sydney

ANTS XVII
Rijksuniversiteit Groningen
10 July 2026

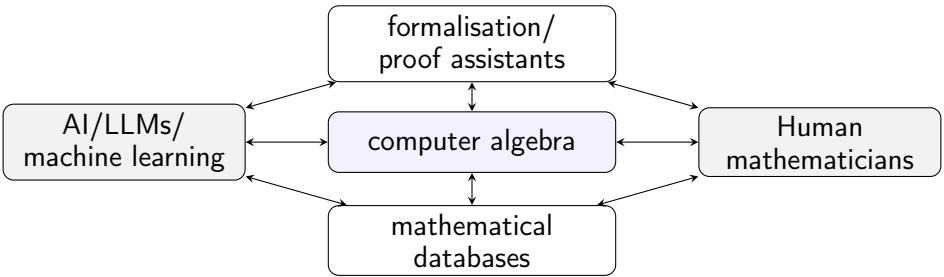
Number theory and a new era for computer algebra

Number theory and a new era for computer algebra

How should our computer algebra systems evolve in an era of large computations, experiments, and new AI-assisted tools?

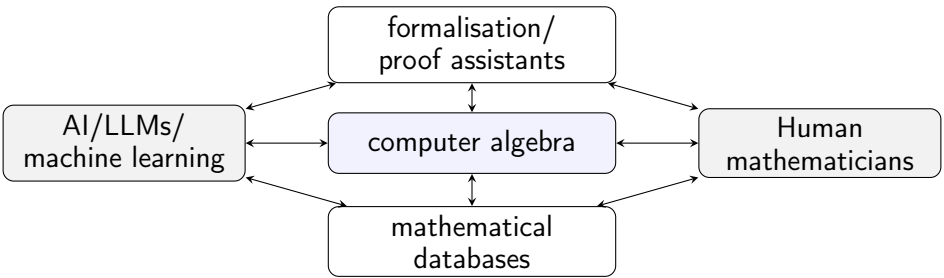
Number theory and a new era for computer algebra

How should our computer algebra systems evolve in an era of large computations, experiments, and new AI-assisted tools?



Number theory and a new era for computer algebra

How should our computer algebra systems evolve in an era of large computations, experiments, and new AI-assisted tools?



Please submit your bugs and feature requests to
https://github.com/magma-maths/magma/
(We'll do our best!)

Magma

Part 1

Ranks of elliptic curves: evidence and statistics

Elliptic curves

Elliptic curves

Let $E\colon y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{Q}$ with $A, B \in \mathbb{Z}$ such that no prime p has $p^4 \mid A$ and $p^6 \mid B$.

Elliptic curves

Let $E\colon y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{Q}$ with $A, B \in \mathbb{Z}$ such that no prime p has $p^4 \mid A$ and $p^6 \mid B$.

The **height** of E is $\operatorname{ht} E = \max(|4A^3|, |27B^2|)$.

Elliptic curves

Let $E\colon y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{Q}$ with $A, B \in \mathbb{Z}$ such that no prime p has $p^4 \mid A$ and $p^6 \mid B$.

The **height** of E is $\operatorname{ht} E = \max(|4A^3|, |27B^2|)$. There are $\asymp H^{5/6} = H^{20/24}$ elliptic curves of height $\leq H$.

Elliptic curves

Let $E\colon y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{Q}$ with $A, B \in \mathbb{Z}$ such that no prime p has $p^4 \mid A$ and $p^6 \mid B$.

The **height** of E is $\operatorname{ht} E = \max(|4A^3|, |27B^2|)$. There are $\asymp H^{5/6} = H^{20/24}$ elliptic curves of height $\leq H$.

Most of the time, the height, conductor $N(E)$, and absolute discriminant $|\Delta(E)| = 16(4A^3 + 27B^2)$ are all close.

Elliptic curves

Let $E\colon y^2 = x^3 + Ax + B$ be an elliptic curve over $\mathbb{Q}$ with $A, B \in \mathbb{Z}$ such that no prime p has $p^4 \mid A$ and $p^6 \mid B$.

The **height** of E is $\operatorname{ht} E = \max(|4A^3|, |27B^2|)$. There are $\asymp H^{5/6} = H^{20/24}$ elliptic curves of height $\leq H$.

Most of the time, the height, conductor $N(E)$, and absolute discriminant $|\Delta(E)| = 16(4A^3 + 27B^2)$ are all close.

For example, $E\colon y^2 = x^3 + 181347x + 893478390$ has

$$\operatorname{ht} E \approx 2 \cdot 10^{19} \quad \textit{and} \quad |\Delta(E)| = N(E) \approx 3 \cdot 10^{20}.$$

Computing the algebraic rank

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent.

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

- $E(\mathbb{Q})/mE(\mathbb{Q}) \simeq E(\mathbb{Q})_{\mathrm{tors}}[m] \oplus (\mathbb{Z}/m\mathbb{Z})^{\mathrm{rk}\, E(\mathbb{Q})}$;

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

- $E(\mathbb{Q})/mE(\mathbb{Q}) \simeq E(\mathbb{Q})_{\mathrm{tors}}[m] \oplus (\mathbb{Z}/m\mathbb{Z})^{\mathrm{rk}\, E(\mathbb{Q})}$;
- $\mathrm{Sel}_m E$ is the m-**Selmer group** of E, a computable finite abelian group;

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

- $E(\mathbb{Q})/mE(\mathbb{Q}) \simeq E(\mathbb{Q})_{\mathrm{tors}}[m] \oplus (\mathbb{Z}/m\mathbb{Z})^{\mathrm{rk}\, E(\mathbb{Q})}$;
- $\mathrm{Sel}_m E$ is the m-**Selmer group** of E, a computable finite abelian group; and
- $\mathrm{Sha}(E)$ is the **Tate-Shafarevich group** of E, a conjecturally finite abelian group.

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

- $E(\mathbb{Q})/mE(\mathbb{Q}) \simeq E(\mathbb{Q})_{\mathrm{tors}}[m] \oplus (\mathbb{Z}/m\mathbb{Z})^{\mathrm{rk}\, E(\mathbb{Q})}$;
- $\mathrm{Sel}_m E$ is the **m-Selmer group** of E, a computable finite abelian group; and
- $\mathrm{Sha}(E)$ is the **Tate-Shafarevich group** of E, a conjecturally finite abelian group.

These algorithms are sped up by assuming:

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

- $E(\mathbb{Q})/mE(\mathbb{Q}) \simeq E(\mathbb{Q})_{\mathrm{tors}}[m] \oplus (\mathbb{Z}/m\mathbb{Z})^{\mathrm{rk}\, E(\mathbb{Q})}$;
- $\mathrm{Sel}_m E$ is the **m-Selmer group** of E, a computable finite abelian group; and
- $\mathrm{Sha}(E)$ is the **Tate-Shafarevich group** of E, a conjecturally finite abelian group.

These algorithms are sped up by assuming:

- Generalized Riemann Hypothesis (GRH) for number fields (used in computing the class group and unit group, more on that later!),

Computing the algebraic rank

We obtain upper and lower bounds on the (algebraic) rank in Magma using descent. For each $m \in \mathbb{Z}_{\geq 1}$, there is an exact sequence

$$0 \to E(\mathbb{Q})/mE(\mathbb{Q}) \to \mathrm{Sel}_m E \to \mathrm{Sha}(E)[m] \to 0;$$

- $E(\mathbb{Q})/mE(\mathbb{Q}) \simeq E(\mathbb{Q})_{\mathrm{tors}}[m] \oplus (\mathbb{Z}/m\mathbb{Z})^{\mathrm{rk}\, E(\mathbb{Q})}$;
- $\mathrm{Sel}_m E$ is the m-**Selmer group** of E, a computable finite abelian group; and
- $\mathrm{Sha}(E)$ is the **Tate-Shafarevich group** of E, a conjecturally finite abelian group.

These algorithms are sped up by assuming:

- Generalized Riemann Hypothesis (GRH) for number fields (used in computing the class group and unit group, more on that later!), and
- Parity conjecture: $(-1)^{\mathrm{rk}\, E(\mathbb{Q})} = w(E)$, where $w(E)$ is the sign of E.

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
After using Cassels-Tate:
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
After using Cassels-Tate:
[...]
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
After using Cassels-Tate:
[...]
After 4-descent:
    0 <= Rank(E) <= 1
    (Z/2)^2 <= Sha(E)[4] <= (Z/4)^1 + (Z/2)^2
(Searched up to height 10^7 on the 4-coverings.)
Time: 7.920
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
After using Cassels-Tate:
[...]
After 4-descent:
    0 <= Rank(E) <= 1
    (Z/2)^2 <= Sha(E)[4] <= (Z/4)^1 + (Z/2)^2
(Searched up to height 10^7 on the 4-coverings.)
Time: 7.920
> bnds;
[ 0, 1 ]
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
After using Cassels-Tate:
[...]
After 4-descent:
    0 <= Rank(E) <= 1
    (Z/2)^2 <= Sha(E)[4] <= (Z/4)^1 + (Z/2)^2
(Searched up to height 10^7 on the 4-coverings.)
Time: 7.920
> bnds;
[ 0, 1 ]
> Sign(LSeries(E));
-1
```

Descent was coded by Geoff Bailey, Steve Donnelly, Mark Watkins, Tom Fisher, and Sebastian Stamminger, with contributions by many others.

The mothership command is `MordellWeilShaInformation` (with many parameters).

Right now, 2- and 4-descent is quite quick, and we are working on improving 3- and 8-descent (which are quite slow).

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([181347,893478390]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 3
After 2-descent:
    0 <= Rank(E) <= 3
    Sha(E)[2] <= (Z/2)^3
(Searched up to height 10000 on the 2-coverings.)
[...]
After using Cassels-Tate:
[...]
After 4-descent:
    0 <= Rank(E) <= 1
    (Z/2)^2 <= Sha(E)[4] <= (Z/4)^1 + (Z/2)^2
(Searched up to height 10^7 on the 4-coverings.)
Time: 7.920
> bnds;
[ 0, 1 ]
> Sign(LSeries(E));
-1
```

So $0 \leq \operatorname{rk} E(\mathbb{Q}) \leq 1$, and by parity $\operatorname{rk} E(\mathbb{Q}) = 1$.

Expectations for the rank

Expectations for the rank

What should we be expecting from the answer? What is the distribution of ranks of elliptic curves over $\mathbb{Q}$?

Expectations for the rank

What should we be expecting from the answer? What is the distribution of ranks of elliptic curves over $\mathbb{Q}$?

Minimalist expectation

Ordered by height, ranks 0 and 1 each occur with density 50%; rank ≥ 2 have density 0.

Expectations for the rank

What should we be expecting from the answer? What is the distribution of ranks of elliptic curves over $\mathbb{Q}$?

Minimalist expectation

Ordered by height, ranks 0 and 1 each occur with density 50%; rank ≥ 2 have density 0.

Unconditionally, the best bound remains the result of Bhargava–Shankar: by height, the average rank is < 0.885, and at least 83.75% of elliptic curves have rank 0 or 1.

Expectations for the rank

What should we be expecting from the answer? What is the distribution of ranks of elliptic curves over $\mathbb{Q}$?

Minimalist expectation

Ordered by height, ranks 0 and 1 each occur with density 50%; rank ≥ 2 have density 0.

Unconditionally, the best bound remains the result of Bhargava–Shankar: by height, the average rank is < 0.885, and at least 83.75% of elliptic curves have rank 0 or 1.

Park–Poonen–V–Wood heuristic

All but finitely many E have $\operatorname{rk} E(\mathbb{Q}) \leq 21$, and for $0 \leq r \leq 20$,

Expectations for the rank

What should we be expecting from the answer? What is the distribution of ranks of elliptic curves over $\mathbb{Q}$?

Minimalist expectation

Ordered by height, ranks 0 and 1 each occur with density 50%; rank ≥ 2 have density 0.

Unconditionally, the best bound remains the result of Bhargava–Shankar: by height, the average rank is < 0.885, and at least 83.75% of elliptic curves have rank 0 or 1.

Park–Poonen–V–Wood heuristic

All but finitely many E have $\operatorname{rk} E(\mathbb{Q}) \leq 21$, and for $0 \leq r \leq 20$,

$$\#\{E : \operatorname{ht} E \leq H \text{ and } \operatorname{rk} E(\mathbb{Q}) \geq r\} = H^{\min(20,21-r)/24+o(1)}.$$

Expectations for the rank

What should we be expecting from the answer? What is the distribution of ranks of elliptic curves over $\mathbb{Q}$?

Minimalist expectation

Ordered by height, ranks 0 and 1 each occur with density 50%; rank ≥ 2 have density 0.

Unconditionally, the best bound remains the result of Bhargava–Shankar: by height, the average rank is < 0.885, and at least 83.75% of elliptic curves have rank 0 or 1.

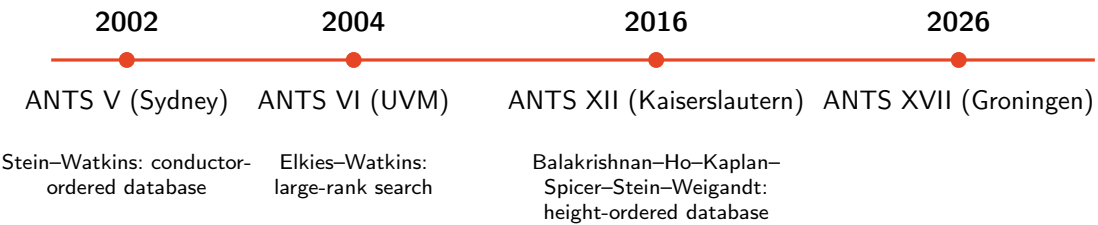
Park–Poonen–V–Wood heuristic

All but finitely many E have $\operatorname{rk} E(\mathbb{Q}) \leq 21$, and for $0 \leq r \leq 20$,

$$\#\{E : \operatorname{ht} E \leq H \text{ and } \operatorname{rk} E(\mathbb{Q}) \geq r\} = H^{\min(20,21-r)/24+o(1)}.$$

For $r = 2$ we predict $H^{19/24}$ (versus $H^{5/6} = H^{20/24}$ for ranks $0, 1$).

Elliptic curve ranks at ANTS



Conductor-ordered data looked too high

Baur Bektemirov, Barry Mazur, William Stein, and Mark Watkins reviewed ranks of elliptic curves using available conductor-ordered data sets.

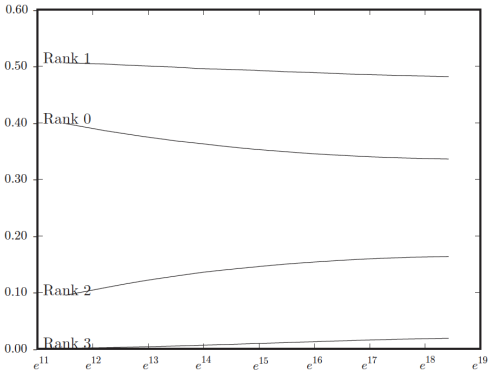
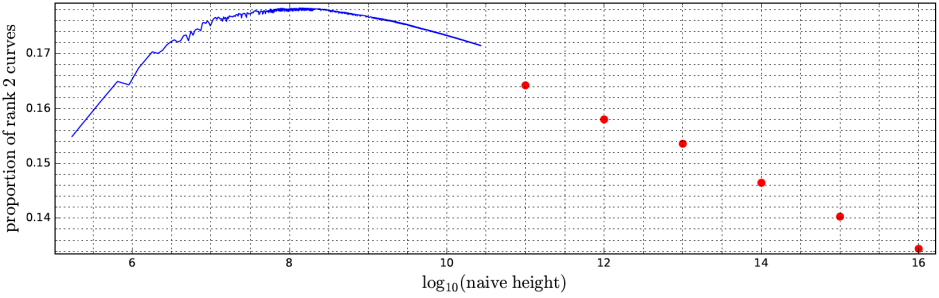


FIGURE 4. Rank Distribution of Stein-Watkins Curves with $N \leq 10^8$

Height-ordered data pushed farther

Jennifer Balakrishnan, Wei Ho, Nathan Kaplan, Simon Spicer, William Stein, and Jamie Weigandt sampled farther, ordered by height.

FIGURE 4. Proportion of rank 2 curves, including samples ($\log_{10}$ scale)



Height-ordered data moves toward the minimalist picture?

Height-ordered data moves toward the minimalist picture?

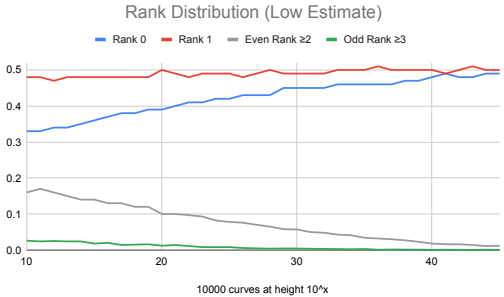
With Blair Butler, using Magma we sampled 10 000 elliptic curves at height $H = 10^n$ for $n = 10, 11, \ldots, 45$.

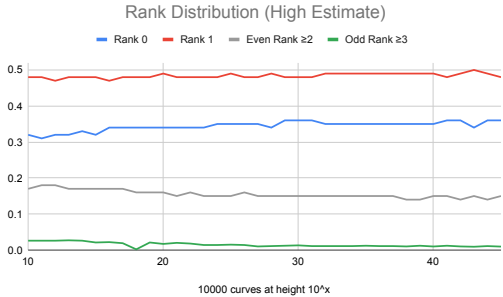
Height-ordered data moves toward the minimalist picture?

With Blair Butler, using Magma we sampled 10 000 elliptic curves at height $H = 10^n$ for $n = 10, 11, \ldots, 45$. The total CPU time was 12 CPU years; wall time was a few weeks.

Height-ordered data moves toward the minimalist picture?

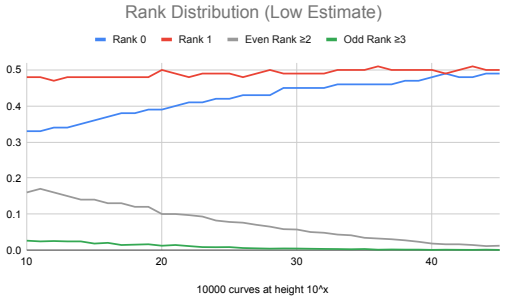
With Blair Butler, using Magma we sampled 10 000 elliptic curves at height $H = 10^n$ for $n = 10, 11, \ldots, 45$. The total CPU time was 12 CPU years; wall time was a few weeks.

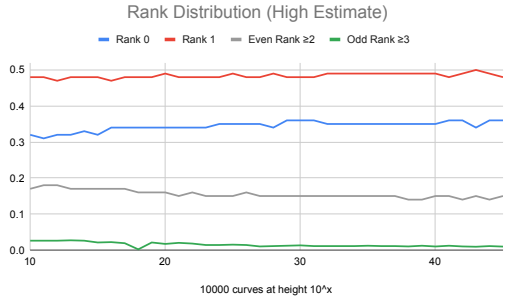




Height-ordered data moves toward the minimalist picture?

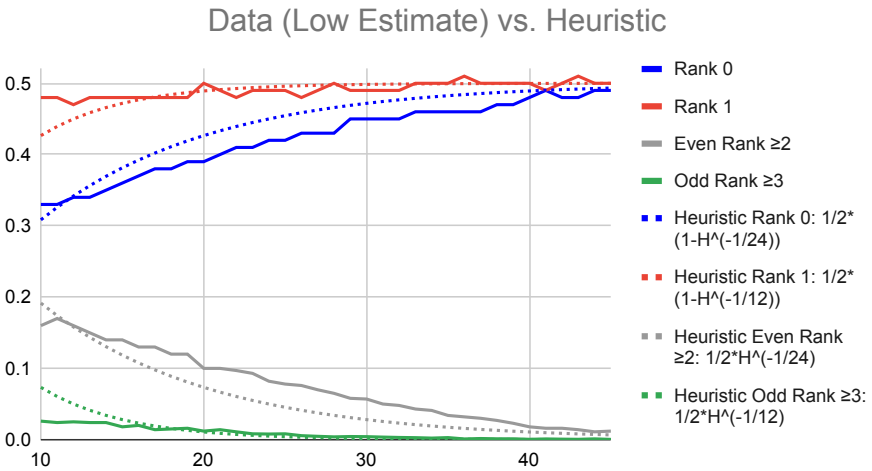
With Blair Butler, using Magma we sampled 10 000 elliptic curves at height $H = 10^n$ for $n = 10, 11, \ldots, 45$. The total CPU time was 12 CPU years; wall time was a few weeks.





The low estimate approaches the expected 50/50 split; the high estimate exposes the remaining ambiguity.

The low estimate tracks the PPVW heuristic



The obstruction is ambiguous rank 0 versus 2

The obstruction is ambiguous rank 0 versus 2

After 4-descent, about 10% of curves remain in the state

$$0 \leq \operatorname{rk} E(\mathbb{Q}) \leq 2,$$

with sign $+1$.

The obstruction is ambiguous rank 0 versus 2

After 4-descent, about 10% of curves remain in the state

$$0 \leq \operatorname{rk} E(\mathbb{Q}) \leq 2,$$

with sign $+1$.

If most of these curves really have rank 0, then the ambiguity is not noise: it is consistent with large 2-primary Tate–Shafarevich groups.

$$(\mathbb{Z}/4\mathbb{Z})^2 \overset{?}{\leq} \operatorname{Sha}(E)[4].$$

The obstruction is ambiguous rank 0 versus 2

After 4-descent, about 10% of curves remain in the state

$$0 \leq \operatorname{rk} E(\mathbb{Q}) \leq 2,$$

with sign $+1$.

If most of these curves really have rank 0, then the ambiguity is not noise: it is consistent with large 2-primary Tate–Shafarevich groups.

$$(\mathbb{Z}/4\mathbb{Z})^2 \overset{?}{\leq} \operatorname{Sha}(E)[4].$$

This is consistent with Delaunay's prediction

$$\operatorname{Prob}\big((\mathbb{Z}/4\mathbb{Z})^2 \subseteq \operatorname{Sha}(E)[2^\infty] \mid \operatorname{rk} E(\mathbb{Q}) = 0\big) \approx 29\%.$$

A typical problem curve: rank 0 or rank 2?

$$E : y^2 = x^3 + 979469x + 631999663.$$

$$0 \leq \mathrm{rk}\, E(\mathbb{Q}) \leq 2.$$

The sign is $+1$, so parity does not remove the ambiguity.

```
> SetClassGroupBounds("GRH");
> E := EllipticCurve([979469, 631999663]);
> time bnds := MordellWeilShaInformation(E);
[...]
The 2-Selmer group has rank 2
After 2-descent:
    0 <= Rank(E) <= 2
    Sha(E)[2] <= (Z/2)^2
[...]
After 4-descent:
    0 <= Rank(E) <= 2
    Sha(E)[4] <= (Z/4)^2
Time: 7.920
> bnds;
[ 0, 2 ]
> Sign(LSeries(E));
1
```

The analytic rank is too expensive to compute directly

The analytic rank is too expensive to compute directly

Let $L(E,s) := \sum_{n\geq 1} a_n/n^s$ and write $N = N(E)$.

The analytic rank is too expensive to compute directly

Let $L(E,s) := \sum_{n\geq 1} a_n/n^s$ and write $N = N(E)$. The Birch–Swinnerton-Dyer conjecture says

$$\operatorname{ord}_{s=1} L(E,s) = \operatorname{rk} E(\mathbb{Q}).$$

The analytic rank is too expensive to compute directly

Let $L(E,s) := \sum_{n\geq 1} a_n/n^s$ and write $N = N(E)$. The Birch–Swinnerton-Dyer conjecture says

$$\operatorname{ord}_{s=1} L(E,s) = \operatorname{rk} E(\mathbb{Q}).$$

Algorithms for computing this order of vanishing take time no better than $\widetilde{O}(\sqrt{N})$.

The analytic rank is too expensive to compute directly

Let $L(E,s) := \sum_{n\geq 1} a_n/n^s$ and write $N = N(E)$. The Birch–Swinnerton-Dyer conjecture says

$$\operatorname{ord}_{s=1} L(E,s) = \operatorname{rk} E(\mathbb{Q}).$$

Algorithms for computing this order of vanishing take time no better than $\widetilde{O}(\sqrt{N})$.

```
> E := EllipticCurve([181347,893478390]);
> time AnalyticRank(E);
```

The analytic rank is too expensive to compute directly

Let $L(E,s) := \sum_{n\geq 1} a_n/n^s$ and write $N = N(E)$. The Birch–Swinnerton-Dyer conjecture says

$$\operatorname{ord}_{s=1} L(E,s) = \operatorname{rk} E(\mathbb{Q}).$$

Algorithms for computing this order of vanishing take time no better than $\widetilde{O}(\sqrt{N})$.

```
> E := EllipticCurve([181347,893478390]);
> time AnalyticRank(E);
```

Joint work with Blair Butler and Edgar Costa

Instead of computing the analytic rank exactly, estimate it with a measure of confidence using the explicit formula.

The explicit formula turns rank into a prime sum

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

$$\operatorname{rk} E(\mathbb{Q})$$

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

$$\operatorname{rk} E(\mathbb{Q}) \overset{\mathrm{BSD}}{=} \operatorname{ord}_{s=1} L(E, s)$$

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

$$\operatorname{rk} E(\mathbb{Q}) \overset{\mathrm{BSD}}{=} \operatorname{ord}_{s=1} L(E,s) \leq Z_\alpha(E) := \sum_{L(E,1+i\gamma)=0} e^{-\alpha\gamma^2}$$

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

$$\operatorname{rk} E(\mathbb{Q}) \overset{\text{BSD}}{=} \operatorname{ord}_{s=1} L(E,s) \le Z_\alpha(E) := \sum_{L(E,1+i\gamma)=0} e^{-\alpha\gamma^2}$$

where $\alpha > 0$ is an **effort** parameter and

$$Z_\alpha(E) = \frac{\log\sqrt{N} - \log(2\pi)}{\sqrt{\alpha\pi}} + C_\Gamma(\alpha) - S_\alpha,$$

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

$$\operatorname{rk} E(\mathbb{Q}) \overset{\text{BSD}}{=} \operatorname{ord}_{s=1} L(E,s) \le Z_\alpha(E) := \sum_{L(E,1+i\gamma)=0} e^{-\alpha\gamma^2}$$

where $\alpha > 0$ is an **effort** parameter and

$$Z_\alpha(E) = \frac{\log\sqrt{N} - \log(2\pi)}{\sqrt{\alpha\pi}} + C_\Gamma(\alpha) - S_\alpha,$$

and where

$$S_\alpha = \frac{1}{\sqrt{\alpha\pi}} \sum_p \log p \sum_{k\ge 1} \frac{a_{p^k}}{p^k} \exp\left(-\frac{\log^2(p^k)}{4\alpha}\right).$$

The explicit formula turns rank into a prime sum

Under GRH for E, the explicit formula with test function $f_\alpha(x) = e^{-\alpha x^2}$ gives

$$\operatorname{rk} E(\mathbb{Q}) \overset{\text{BSD}}{=} \operatorname{ord}_{s=1} L(E,s) \le Z_\alpha(E) := \sum_{L(E,1+i\gamma)=0} e^{-\alpha\gamma^2}$$

where $\alpha > 0$ is an **effort** parameter and

$$Z_\alpha(E) = \frac{\log\sqrt{N} - \log(2\pi)}{\sqrt{\alpha\pi}} + C_\Gamma(\alpha) - S_\alpha,$$

and where

$$S_\alpha = \frac{1}{\sqrt{\alpha\pi}} \sum_p \log p \sum_{k\ge 1} \frac{a_{p^k}}{p^k} \exp\left(-\frac{\log^2(p^k)}{4\alpha}\right).$$

As the effort $\alpha \to \infty$, the right side converges to $r = \operatorname{rk} E(\mathbb{Q})$.

Unweighted prime sums converge noisily

Unweighted prime sums converge noisily

We focus on the prime part and consider weighted sums

$$\sum_{p \leq B} \frac{a_p \log p}{p} W(\log p).$$

Unweighted prime sums converge noisily

We focus on the prime part and consider weighted sums

$$\sum_{p \le B} \frac{a_p \log p}{p} W(\log p).$$

For $W = 1$, Kim–Murty show that if the limit exists,

$$\frac{1}{\log B} \sum_{p \le B} \frac{a_p \log p}{p} \sim -r + 1/2.$$

Unweighted prime sums converge noisily

We focus on the prime part and consider weighted sums

$$\sum_{p \le B} \frac{a_p \log p}{p} W(\log p).$$

For $W = 1$, Kim–Murty show that if the limit exists,

$$\frac{1}{\log B} \sum_{p \le B} \frac{a_p \log p}{p} \sim -r + 1/2.$$

Problem

The variance is constant $1/2$.

Unweighted prime sums converge noisily

We focus on the prime part and consider weighted sums

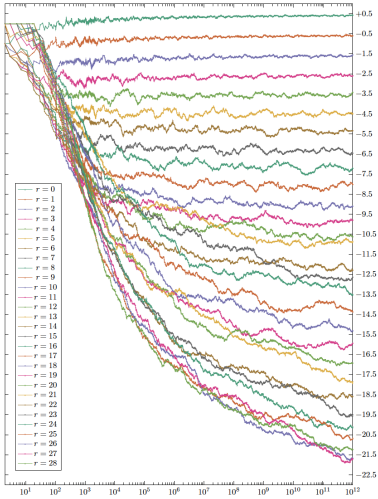
$$\sum_{p \le B} \frac{a_p \log p}{p} W(\log p).$$

For $W = 1$, Kim–Murty show that if the limit exists,

$$\frac{1}{\log B} \sum_{p \le B} \frac{a_p \log p}{p} \sim -r + 1/2.$$

Problem

The variance is constant $1/2$.



Computed by Drew Sutherland

Compact support makes rigorous bounds expensive

Compact support makes rigorous bounds expensive

The standard test function, used by Bober, yields

$$W_\Delta(u) = \frac{1}{\pi\Delta} \max\left(1 - \frac{|u|}{2\pi\Delta}, 0\right),$$

Compact support makes rigorous bounds expensive

The standard test function, used by Bober, yields

$$W_\Delta(u) = \frac{1}{\pi\Delta} \max\left(1 - \frac{|u|}{2\pi\Delta}, 0\right),$$

so the prime sum has compact

Compact support makes rigorous bounds expensive

The standard test function, used by Bober, yields

$$W_\Delta(u) = \frac{1}{\pi\Delta} \max\left(1 - \frac{|u|}{2\pi\Delta}, 0\right),$$

so the prime sum has compact but exponential support $\log p \leq 2\pi\Delta$.

Compact support makes rigorous bounds expensive

The standard test function, used by Bober, yields

$$W_\Delta(u) = \frac{1}{\pi\Delta} \max\left(1 - \frac{|u|}{2\pi\Delta}, 0\right),$$

so the prime sum has compact but exponential support $\log p \leq 2\pi\Delta$.

```
analytic_rank_upper_bound(max_Delta=None, adaptive=True, N=None,
    root_number='compute', bad_primes=None, ncpus=None)          [source]
```

> **Warning**
>
> Zero sum computation time is exponential in the tightness parameter Δ, roughly doubling for every increase of 0.1 thereof. Using $\Delta = 1$ (and adaptive=False) will yield a runtime of a few milliseconds; $\Delta = 2$ takes a few seconds, and $\Delta = 3$ may take upwards of an hour. Increase beyond this at your own risk!

Sato–Tate predicts the tail variance

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$.

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$. The Sato–Tate distribution gives

$$\mathbb{E}(\lambda_p) = 0, \qquad \mathrm{Var}(\lambda_p) = 1.$$

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$. The Sato–Tate distribution gives

$$\mathbb{E}(\lambda_p) = 0, \qquad \mathrm{Var}(\lambda_p) = 1.$$

For an interval $[A, B]$, the model gives

$$\mathrm{Var}\left(\frac{1}{\sqrt{\pi\alpha}} \sum_{A<p\le B} \lambda_p \frac{\log p}{\sqrt{p}} e^{-\log^2 p/(4\alpha)} \right)$$

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$. The Sato–Tate distribution gives

$$\mathbb{E}(\lambda_p) = 0, \qquad \mathrm{Var}(\lambda_p) = 1.$$

For an interval $[A, B]$, the model gives

$$\mathrm{Var}\left(\frac{1}{\sqrt{\pi\alpha}} \sum_{A<p\le B} \lambda_p \frac{\log p}{\sqrt{p}} e^{-\log^2 p/(4\alpha)}\right) \approx \frac{1}{\pi\alpha} \int_A^B \frac{\log^2 x}{x} \frac{1}{\log x} e^{-\log^2 x/(2\alpha)}\, \mathrm{d}x$$

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$. The Sato–Tate distribution gives

$$\mathbb{E}(\lambda_p) = 0, \qquad \mathrm{Var}(\lambda_p) = 1.$$

For an interval $[A, B]$, the model gives

$$\begin{aligned}\mathrm{Var}\left(\frac{1}{\sqrt{\pi\alpha}} \sum_{A<p\le B} \lambda_p \frac{\log p}{\sqrt{p}} e^{-\log^2 p/(4\alpha)}\right) &\approx \frac{1}{\pi\alpha}\int_A^B \frac{\log^2 x}{x}\frac{1}{\log x} e^{-\log^2 x/(2\alpha)}\,\mathrm{d}x \\ &= \frac{1}{\pi\alpha}\int_A^B \frac{\log x}{x} e^{-\log^2 x/(2\alpha)}\,\mathrm{d}x\end{aligned}$$

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$. The Sato–Tate distribution gives

$$\mathbb{E}(\lambda_p) = 0, \qquad \mathrm{Var}(\lambda_p) = 1.$$

For an interval $[A, B]$, the model gives

$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{\sqrt{\pi\alpha}} \sum_{A<p\le B} \lambda_p \frac{\log p}{\sqrt{p}} e^{-\log^2 p/(4\alpha)}\right) &\approx \frac{1}{\pi\alpha} \int_A^B \frac{\log^2 x}{x} \frac{1}{\log x} e^{-\log^2 x/(2\alpha)}\,\mathrm{d}x \\
&= \frac{1}{\pi\alpha} \int_A^B \frac{\log x}{x} e^{-\log^2 x/(2\alpha)}\,\mathrm{d}x \\
&= \frac{1}{\pi}\left(e^{-\log^2 A/(2\alpha)} - e^{-\log^2 B/(2\alpha)}\right).
\end{aligned}$$

Sato–Tate predicts the tail variance

Let $\lambda_p = a_p/\sqrt{p}$. The Sato–Tate distribution gives

$$\mathbb{E}(\lambda_p) = 0, \qquad \mathrm{Var}(\lambda_p) = 1.$$

For an interval $[A, B]$, the model gives

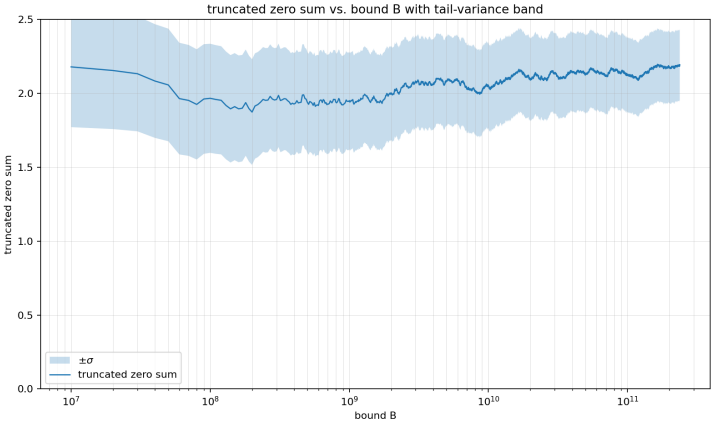
$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{\sqrt{\pi\alpha}} \sum_{A<p\le B} \lambda_p \frac{\log p}{\sqrt{p}} e^{-\log^2 p/(4\alpha)}\right) &\approx \frac{1}{\pi\alpha} \int_A^B \frac{\log^2 x}{x} \frac{1}{\log x} e^{-\log^2 x/(2\alpha)}\,\mathrm{d}x \\
&= \frac{1}{\pi\alpha} \int_A^B \frac{\log x}{x} e^{-\log^2 x/(2\alpha)}\,\mathrm{d}x \\
&= \frac{1}{\pi}\left(e^{-\log^2 A/(2\alpha)} - e^{-\log^2 B/(2\alpha)}\right).
\end{aligned}$$

Using a Gaussian for a rank-2 curve

Using a Gaussian for a rank-2 curve

For $\alpha = 100$, plot $Z_\alpha(E)$ truncated up to B for

$$E : y^2 = x^3 + 7351286338x + 167280101764500.$$



Rejecting rank 2 under a variance model

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*.

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*. Truncating at B leaves missing rank bias

$$r\xi := r \operatorname{erfc}\left(\frac{\log B}{2\sqrt{\alpha}}\right).$$

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*. Truncating at B leaves missing rank bias

$$r\xi := r \operatorname{erfc}\left(\frac{\log B}{2\sqrt{\alpha}}\right).$$

Problem curve example

With $\alpha = 40$ and $B = 10^7$, we compute $Z_\alpha(E) = 1.39208$ in 34 seconds.

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*. Truncating at B leaves missing rank bias

$$r\xi := r \operatorname{erfc}\left(\frac{\log B}{2\sqrt{\alpha}}\right).$$

Problem curve example

With $\alpha = 40$ and $B = 10^7$, we compute $Z_\alpha(E) = 1.39208$ in 34 seconds. The conservative threshold test for rank 2 gives

$$Z_\alpha(E) + r\xi + c\sigma = 1.39208 + 2 \cdot 0.0628 + c \cdot 0.1 < 2$$

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*. Truncating at B leaves missing rank bias

$$r\xi := r \operatorname{erfc}\left(\frac{\log B}{2\sqrt{\alpha}}\right).$$

Problem curve example

With $\alpha = 40$ and $B = 10^7$, we compute $Z_\alpha(E) = 1.39208$ in 34 seconds. The conservative threshold test for rank 2 gives

$$Z_\alpha(E) + r\xi + c\sigma = 1.39208 + 2 \cdot 0.0628 + c \cdot 0.1 < 2$$

for $c = 4.8$.

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*. Truncating at B leaves missing rank bias

$$r\xi := r \operatorname{erfc}\left(\frac{\log B}{2\sqrt{\alpha}}\right).$$

Problem curve example

With $\alpha = 40$ and $B = 10^7$, we compute $Z_\alpha(E) = 1.39208$ in 34 seconds. The conservative threshold test for rank 2 gives

$$Z_\alpha(E) + r\xi + c\sigma = 1.39208 + 2 \cdot 0.0628 + c \cdot 0.1 < 2$$

for $c = 4.8$. Rank 2 is a 4.8σ deviation.

Rejecting rank 2 under a variance model

The Kim–Murty asymptotic and partial summation give a model for the rank-dependent *mean*. Truncating at B leaves missing rank bias

$$r\xi := r \operatorname{erfc}\left(\frac{\log B}{2\sqrt{\alpha}}\right).$$

Problem curve example

With $\alpha = 40$ and $B = 10^7$, we compute $Z_\alpha(E) = 1.39208$ in 34 seconds. The conservative threshold test for rank 2 gives

$$Z_\alpha(E) + r\xi + c\sigma = 1.39208 + 2 \cdot 0.0628 + c \cdot 0.1 < 2$$

for $c = 4.8$. Rank 2 is a 4.8σ deviation.

Similarly, the log likelihood ratio is 16.9 in favor of rank 0 under equal priors.

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.
- For the remaining 10%, which are of rank 0 or rank 2, we can instead apply an *analytic rank estimator* for a statistical test.

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.
- For the remaining 10%, which are of rank 0 or rank 2, we can instead apply an *analytic rank estimator* for a statistical test. It is embarrassingly parallelizable!

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.
- For the remaining 10%, which are of rank 0 or rank 2, we can instead apply an *analytic rank estimator* for a statistical test. It is embarrassingly parallelizable!
- The best accuracy comes from fitting to a sum of shifted Gaussians $f_{\alpha,t}(x)$!

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.
- For the remaining 10%, which are of rank 0 or rank 2, we can instead apply an *analytic rank estimator* for a statistical test. It is embarrassingly parallelizable!
- The best accuracy comes from fitting to a sum of shifted Gaussians $f_{\alpha,t}(x)$!
- We have also seen speedups using *sampling* to extend the range:

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.
- For the remaining 10%, which are of rank 0 or rank 2, we can instead apply an *analytic rank estimator* for a statistical test. It is embarrassingly parallelizable!
- The best accuracy comes from fitting to a sum of shifted Gaussians $f_{\alpha,t}(x)$!
- We have also seen speedups using *sampling* to extend the range: if a sum has variance σ^2 and you estimate it by sampling without replacement a proportion $1/k$, the standard deviation of the estimator is $\sigma\sqrt{k-1}$.

Rank computations: takeaways

- On an elliptic curve of large height, assuming GRH, descent determines the algebraic rank for about 90% of curves.
- For the remaining 10%, which are of rank 0 or rank 2, we can instead apply an *analytic rank estimator* for a statistical test. It is embarrassingly parallelizable!
- The best accuracy comes from fitting to a sum of shifted Gaussians $f_{\alpha,t}(x)$!
- We have also seen speedups using *sampling* to extend the range: if a sum has variance σ^2 and you estimate it by sampling without replacement a proportion $1/k$, the standard deviation of the estimator is $\sigma\sqrt{k-1}$.

Question

What do you want to see from this type of (statistical) algorithm inside a computer algebra system?

Part 2

Class groups of number fields: a few anecdotes

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Let K be a number field with ring of integers $\mathbb{Z}_K$.

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Let K be a number field with ring of integers $\mathbb{Z}_K$. To compute the class group $\mathrm{Cl}\,\mathbb{Z}_K$, we use the *factor base* method (generators modulo relations).

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Let K be a number field with ring of integers $\mathbb{Z}_K$. To compute the class group $\mathrm{Cl}\,\mathbb{Z}_K$, we use the *factor base* method (generators modulo relations).

Let $S := \{\mathfrak{p}_1, \ldots, \mathfrak{p}_m\}$ be a set of primes whose classes generate the class group.

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Let K be a number field with ring of integers $\mathbb{Z}_K$. To compute the class group $\mathrm{Cl}\,\mathbb{Z}_K$, we use the *factor base* method (generators modulo relations).

Let $S := \{\mathfrak{p}_1, \ldots, \mathfrak{p}_m\}$ be a set of primes whose classes generate the class group. Let

$$\mathbb{Z}_K[S^{-1}]^\times := \{\alpha \in K^\times : \mathrm{ord}_\mathfrak{p}(\alpha) = 0 \text{ if } \mathfrak{p} \notin S\}$$

be the group of S-units in $\mathbb{Z}_K$.

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Let K be a number field with ring of integers $\mathbb{Z}_K$. To compute the class group $\mathrm{Cl}\,\mathbb{Z}_K$, we use the *factor base* method (generators modulo relations).

Let $S := \{\mathfrak{p}_1, \ldots, \mathfrak{p}_m\}$ be a set of primes whose classes generate the class group. Let

$$\mathbb{Z}_K[S^{-1}]^\times := \{\alpha \in K^\times : \mathrm{ord}_\mathfrak{p}(\alpha) = 0 \text{ if } \mathfrak{p} \notin S\}$$

be the group of S-units in $\mathbb{Z}_K$. Then there is an exact sequence

$$\begin{aligned} 1 \to \mathbb{Z}_K^\times \to \mathbb{Z}_K[S^{-1}]^\times &\xrightarrow{\phi} \mathbb{Z}^m \xrightarrow{[\,]} \mathrm{Cl}\,\mathbb{Z}_K \to 1 \\ \alpha &\mapsto (\mathrm{ord}_{\mathfrak{p}_i}(\alpha))_i \end{aligned}$$

Class groups reduce to generators and relations

Joint work with Stephan Elsenhans.

Let K be a number field with ring of integers $\mathbb{Z}_K$. To compute the class group $\mathrm{Cl}\,\mathbb{Z}_K$, we use the *factor base* method (generators modulo relations).

Let $S := \{\mathfrak{p}_1, \ldots, \mathfrak{p}_m\}$ be a set of primes whose classes generate the class group. Let

$$\mathbb{Z}_K[S^{-1}]^\times := \{\alpha \in K^\times : \mathrm{ord}_\mathfrak{p}(\alpha) = 0 \text{ if } \mathfrak{p} \notin S\}$$

be the group of S-units in $\mathbb{Z}_K$. Then there is an exact sequence

$$\begin{aligned} 1 \to \mathbb{Z}_K^\times \to \mathbb{Z}_K[S^{-1}]^\times &\xrightarrow{\phi} \mathbb{Z}^m \xrightarrow{[\,]} \mathrm{Cl}\,\mathbb{Z}_K \to 1 \\ \alpha &\mapsto (\mathrm{ord}_{\mathfrak{p}_i}(\alpha))_i \end{aligned}$$

We search for enough relations in $\mathbb{Z}_K[S^{-1}]$ to generate the kernel of the class map; we obtain the unit group from $\ker \phi$.

Rigorous method via saturation

Rigorous method via saturation

Intended default behavior

`ClassGroup` uses the Minkowski bound for generators, finds enough relations to get a finite group, then certifies with an *exact saturation*.

Rigorous method via saturation

Intended default behavior

`ClassGroup` uses the Minkowski bound for generators, finds enough relations to get a finite group, then certifies with an *exact saturation*.

Bug found and fixed

A class group saturation check had been switched off. The result was therefore not rigorous, though wrong answers were unlikely in ordinary use.

This bug has been fixed in V2.29.

Rigorous method via saturation

Intended default behavior

`ClassGroup` uses the Minkowski bound for generators, finds enough relations to get a finite group, then certifies with an *exact saturation*.

Bug found and fixed

A class group saturation check had been switched off. The result was therefore not rigorous, though wrong answers were unlikely in ordinary use.

This bug has been fixed in V2.29.

We added stress tests where relation search stops early after full rank appears, but before the analytic class number formula matches.

Conditional algorithms with stopping criterion

Conditional algorithms with stopping criterion

If you run `SetClassGroupBounds("GRH")`, then a much smaller norm bound on generators can be used: Bach, Belabas–Diaz y Diaz–Friedman.

Conditional algorithms with stopping criterion

If you run `SetClassGroupBounds("GRH")`, then a much smaller norm bound on generators can be used: Bach, Belabas–Diaz y Diaz–Friedman.

A natural stopping signal comes from the analytic class number formula:

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \frac{2^r (2\pi)^c h R}{w\sqrt{|d|}}.$$

Conditional algorithms with stopping criterion

If you run `SetClassGroupBounds("GRH")`, then a much smaller norm bound on generators can be used: Bach, Belabas–Diaz y Diaz–Friedman.

A natural stopping signal comes from the analytic class number formula:

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \frac{2^r(2\pi)^c hR}{w\sqrt{|d|}}.$$

One can evaluate $\zeta_K^*(1)$ rigorously to precision ϵ using weighted sums, and similarly with the purported regulator.

Conditional algorithms with stopping criterion

If you run `SetClassGroupBounds("GRH")`, then a much smaller norm bound on generators can be used: Bach, Belabas–Diaz y Diaz–Friedman.

A natural stopping signal comes from the analytic class number formula:

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \frac{2^r(2\pi)^c hR}{w\sqrt{|d|}}.$$

One can evaluate $\zeta_K^*(1)$ rigorously to precision ϵ using weighted sums, and similarly with the purported regulator.

Bug found and fixed

A short Euler product evaluation was used in a place where a rigorous estimate was required. The result was therefore not rigorous.

This bug has been fixed in V2.29.

The Euler product is far better than worst-case bounds

The Euler product is far better than worst-case bounds

For the Dedekind zeta residue,

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \prod_p \frac{1 - 1/p}{\prod_{\mathfrak{p}|p}(1 - 1/\mathrm{N}\,\mathfrak{p})}.$$

The Euler product is far better than worst-case bounds

For the Dedekind zeta residue,

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \prod_p \frac{1 - 1/p}{\prod_{\mathfrak{p}\mid p}(1 - 1/\mathrm{N}\,\mathfrak{p})}.$$

Two facts in tension

- The product converges, but rigorous bounds can be far too pessimistic for practical stopping.

The Euler product is far better than worst-case bounds

For the Dedekind zeta residue,

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \prod_p \frac{1 - 1/p}{\prod_{\mathfrak{p}|p}(1 - 1/\mathrm{N}\,\mathfrak{p})}.$$

Two facts in tension

- The product converges, but rigorous bounds can be far too pessimistic for practical stopping.
- The direct truncated Euler product is remarkably accurate in experiments.

The Euler product is far better than worst-case bounds

For the Dedekind zeta residue,

$$\zeta_K^*(1) = \operatorname*{res}_{s=1} \zeta_K(s) = \prod_p \frac{1 - 1/p}{\prod_{\mathfrak{p}|p}(1 - 1/\mathrm{N}\,\mathfrak{p})}.$$

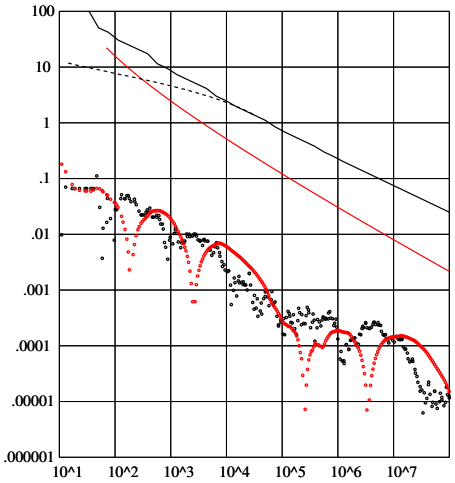
Two facts in tension

- The product converges, but rigorous bounds can be far too pessimistic for practical stopping.
- The direct truncated Euler product is remarkably accurate in experiments.

Across thousands of polynomials, the ratio of the product truncated at $p < 1000$ to the GRH residue had mean 1.012 and sample standard deviation 0.015.

Safe bounds can be pessimistic by orders of magnitude

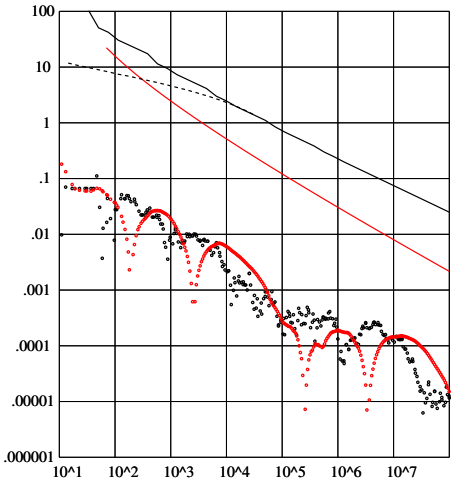
Safe bounds can be pessimistic by orders of magnitude



Legend

- black: direct Euler product and Bach-style bound;
- red: weighted sum and Belabas–Friedman bound;
- dots: observed errors;
- lines: proved or conditional bounds.

Safe bounds can be pessimistic by orders of magnitude



Legend

- black: direct Euler product and Bach-style bound;
- red: weighted sum and Belabas–Friedman bound;
- dots: observed errors;
- lines: proved or conditional bounds.

Typical errors are much smaller!

Class group computations: takeaways

Class group computations: takeaways

- Heuristic stopping is useful: the Euler product is much better in practice than worst-case bounds suggest.

Class group computations: takeaways

- Heuristic stopping is useful: the Euler product is much better in practice than worst-case bounds suggest.
- Reliability requires robust testing targeted at key steps, not only at the main search routine.

Class group computations: takeaways

- Heuristic stopping is useful: the Euler product is much better in practice than worst-case bounds suggest.
- Reliability requires robust testing targeted at key steps, not only at the main search routine.

Question

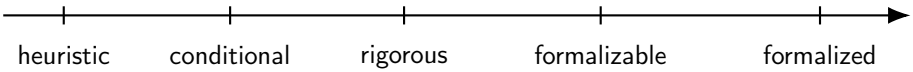
How do we ensure that computer algebra systems are reliable and rigorous, meeting users' expectations?

Part 3

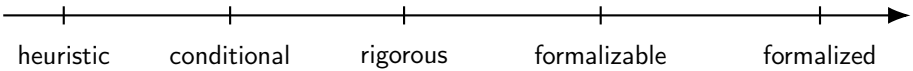
Rigor and reliability in computer algebra

Rigor axis

Rigor axis

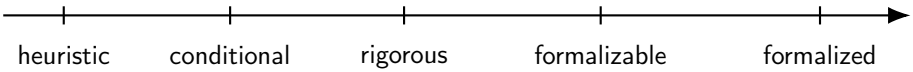

heuristic
conditional
rigorous
formalizable
formalized

Rigor axis



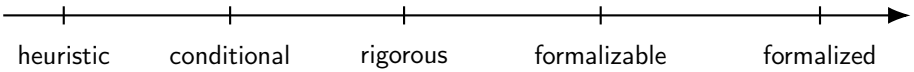
regime	meaning	example
heuristic	allowed to guess, estimate, or truncate	short Euler product

Rigor axis



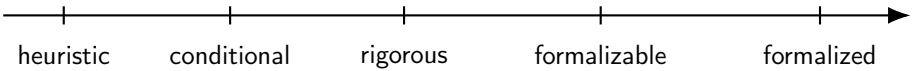
regime	meaning	example
heuristic	allowed to guess, estimate, or truncate	short Euler product
conditional	assumes a theorem such as GRH	GRH-conditional class number

Rigor axis



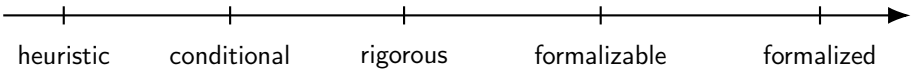
regime	meaning	example
heuristic	allowed to guess, estimate, or truncate	short Euler product
conditional	assumes a theorem such as GRH	GRH-conditional class number
rigorous	proof with numerical error analysis	regulator bounds

Rigor axis



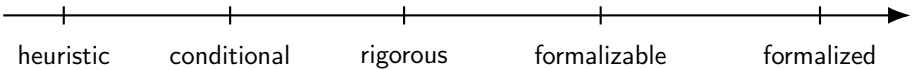
regime	meaning	example
heuristic	allowed to guess, estimate, or truncate	short Euler product
conditional	assumes a theorem such as GRH	GRH-conditional class number
rigorous	proof with numerical error analysis	regulator bounds
formalizable	exact proof object could be checked	class group with saturation

Rigor axis



regime	meaning	example
heuristic	allowed to guess, estimate, or truncate	short Euler product
conditional	assumes a theorem such as GRH	GRH-conditional class number
rigorous	proof with numerical error analysis	regulator bounds
formalizable	exact proof object could be checked	class group with saturation
formalized	checked by a proof assistant	Lean theorem

Rigor axis



regime	meaning	example
heuristic	allowed to guess, estimate, or truncate	short Euler product
conditional	assumes a theorem such as GRH	GRH-conditional class number
rigorous	proof with numerical error analysis	regulator bounds
formalizable	exact proof object could be checked	class group with saturation
formalized	checked by a proof assistant	Lean theorem

Different users may expect different proof levels from the same command.

What are your expectations for rigor from these functions?

- `IsPrime`

What are your expectations for rigor from these functions?

- `IsPrime`
- `ClassGroup`

What are your expectations for rigor from these functions?

- `IsPrime`
- `ClassGroup`
- `GaloisGroup`

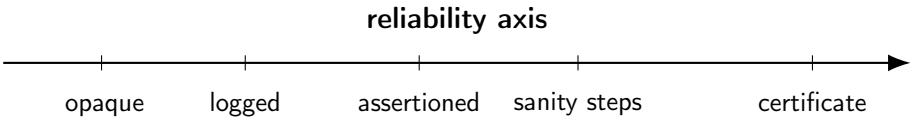
What are your expectations for rigor from these functions?

- `IsPrime`
- `ClassGroup`
- `GaloisGroup`
- `GroebnerBasis`

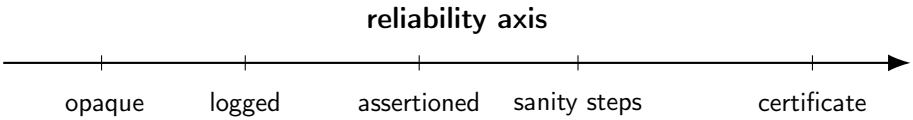
What are your expectations for rigor from these functions?

- `IsPrime`
- `ClassGroup`
- `GaloisGroup`
- `GroebnerBasis`
- `LLL`

Reliability

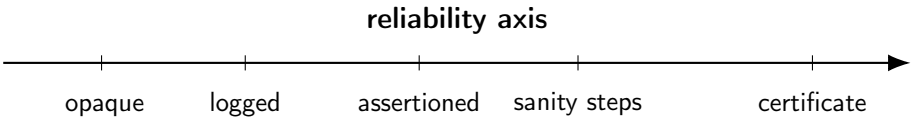


Reliability



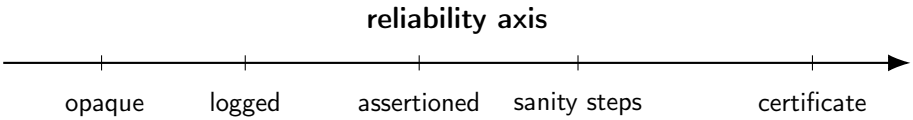
regime	meaning
opaque	just the answer

Reliability



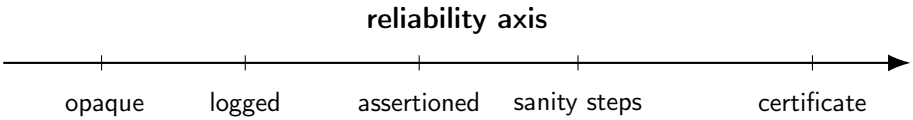
regime	meaning
opaque	just the answer
logged	verbose output

Reliability



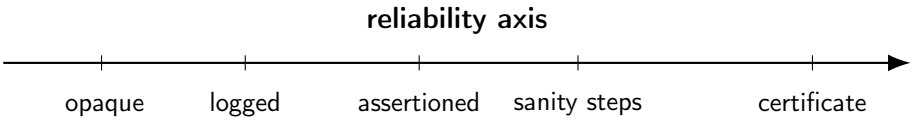
regime	meaning
opaque	just the answer
logged	verbose output
assertioned	checks along the way

Reliability



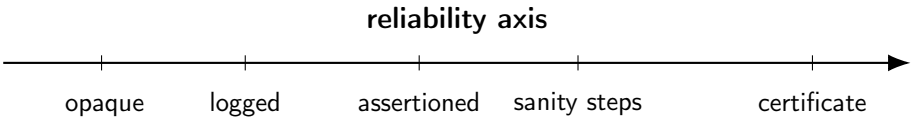
regime	meaning
opaque	just the answer
logged	verbose output
assertioned	checks along the way
sanity steps	results of intermediate computation that could be independently checked

Reliability



regime	meaning
opaque	just the answer
logged	verbose output
assertioned	checks along the way
sanity steps	results of intermediate computation that could be independently checked
certificate	enough data to formalize

Reliability



regime	meaning
opaque	just the answer
logged	verbose output
assertioned	checks along the way
sanity steps	results of intermediate computation that could be independently checked
certificate	enough data to formalize

(Maybe some day we will formalize the algorithms themselves?)

Computer algebra in the future

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

- stable APIs/MCPs for typed mathematical objects?

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

- stable APIs/MCPs for typed mathematical objects?
- explicit proof levels in outputs?

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

- stable APIs/MCPs for typed mathematical objects?
- explicit proof levels in outputs?
- certificate checkers for expensive algorithms?

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

- stable APIs/MCPs for typed mathematical objects?
- explicit proof levels in outputs?
- certificate checkers for expensive algorithms?
- reproducible logs, seeds, and provenance?

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

- stable APIs/MCPs for typed mathematical objects?
- explicit proof levels in outputs?
- certificate checkers for expensive algorithms?
- reproducible logs, seeds, and provenance?
- benchmark suites designed for real mathematical workloads?

Computer algebra in the future

Agents are calling computer algebra systems constantly. They mix exact algebra, databases, numerical routines, and proof assistants.

What should we build?

- stable APIs/MCPs for typed mathematical objects?
- explicit proof levels in outputs?
- certificate checkers for expensive algorithms?
- reproducible logs, seeds, and provenance?
- benchmark suites designed for real mathematical workloads?

Possible proposal: output from a computer algebra system is a *type*, a mathematical claim with metadata that can include assumptions, confidence, reliability provenance, and a path to verification.