Release notes for Magma V2.29

September 2025

Contents

1 Introduction **3**

2 Documentation **3**

3 Bugs fixed since V2.28-27 **3**

4 Large Version **4**

5 Aggregates and Mappings **5**
5.1 Associative Arrays 5

6 Algebraic Geometry **5**
6.1 Schemes 5
6.2 Algebraic Curves 5

7 Arithmetic Geometry **6**
7.1 Hyperelliptic Curves and Jacobians 6
7.2 Modular Symbols 6
7.3 Algebraic Modular Forms (New) 6

8 Arithmetic Fields (Global) **7**
8.1 Algebraic Number Fields 7
8.1.1 Quadratic Fields 7
8.2 Algebraic Function Fields 8
8.3 Galois Groups 8

9 Arithmetic Fields (Local) **9**
9.1 p-adic Rings and their Extensions 9
9.2 Newton Polygons 9

10 Basic Rings and Fields **9**
10.1 Integer Ring 9

11 Commutative Algebra **9**
11.1 Univariate Polynomial Rings 9
11.2 Multivariate Polynomial Rings 10
11.3 Ideal Theory and Gröbner Bases 10

12 Groups **10**
12.1 Reductive Groups (New) 10

13 Language and System Features **10**
13.1 Language 10

14 Lattices and Quadratic Forms **11**
14.1 Lattices 11

15 Lie Theory **12**
15.1 Representation Theory 12

16 Linear Algebra and Module Theory **12**
16.1 Matrices 12
16.2 Sparse Matrices 12

17 Linear Associative Algebras **12**
17.1 Matrix Algebras 12
17.2 Étale Algebras 13
17.3 Finitely Presented Associative Algebras 13

18 Representation Theory **13**
18.1 Character Theory 13
18.2 Group Representations (New) 14

1 Introduction

This document provides a terse summary of the new features released as part of Magma versions V2.29 (September 2025).

A small number of new features were exported in patch releases prior to the main release of V2.29 in September 2025 and these are also listed here for completeness. For a complete list of bugfixes throughout V2.28, the reader should consult the patch release change logs for V2.28-n for n from 2 to 27.

2 Documentation

New Handbook Chapters are the following:

- Reductive Groups (75)
- Group Representations (102)
- Algebraic Modular Forms (150)

3 Bugs fixed since V2.28-27

- A crash in `Embed` for finite fields (involving default and non-default fields) has been fixed. Reported by Sachi Hashimoto.
- A bug in NewformDecomposition involving a class of newforms over number fields has been fixed. Reported by Begum Cakti.
- Incorrect documentation for the function Discriminant for univariate polynomials has been fixed. Reported by Michael Zieve.

4 Large Version

A new '**Large**' version of Magma has been developed, which is available as an optional executable download (currently for the 64-bit Linux `avx2` version only).

The existing default '**Standard**' 64-bit version of Magma (which will remain as the default version) has always used 64-bit operations for operations involving packed data (including large integers, matrices over small finite fields, permutations, etc.) and the total memory allowable or the size of any individual memory block has been fully 64-bit (with no 32-bit limit). But the Standard version uses 32-bit quantities for indexing generic Magma objects, so this means that it has certain limits on the **number** of generic Magma objects or the **size** of some types of Magma aggregate structures (such as sets and sequences).

In contrast, the new Large version of Magma uses 64-bit quantities in such contexts. Thus the following have 64-bit limits in the Large version (instead of 32-bit limits in the Standard version):

1. The number of memory blocks (in the Standard version, if that number is reached, Magma must abort immediately). In particular, this includes the number of large integers (integers which must be stored in memory blocks) which can be created.

2. The cardinality of sets, sequences and other aggregate structures.

3. The length of objects represented as vectors (including standard vectors over rings).

4. The number of rows/columns in matrices (both dense and sparse).

5. The size of the matrices in the linear algebra phase of the F_4 Gröbner basis algorithm.

At the same time, 64-bit long integers are exploited more in the Large version in algorithms which use relatively small integers. In particular:

1. Integers with absolute value between 2^{30} and 2^{62} are not stored in a separate block (but become 'immediate' small [machine 64-bit long] integers), thus saving time and space for computations with integers in this range. A similar improvement is achieved for elements of finite fields with cardinality in this range, and similar types using integers in this range.

2. Algorithms for matrices or polynomials with entries from the ring of integers or a finite field which are up to 64-bits are sped up similarly.

PLEASE NOTE: the Large version may need significantly more total memory than the Standard version for many kinds of computations (because the indexing of unpacked generic Magma objects is changed from using 32-bit to 64-bit quantities), so that is why the Standard version remains the default for now. But the Large version will allow Magma to perform several types of computations which have been impossible up till now, and it overcomes several errors/failures which have been arising when 32-bit quantities overflow in the Standard version.

5 Aggregates and Mappings

5.1 Associative Arrays

New Features:

– When creating an associative array A, one may now specify a default value D (any constant value) to be associated with A, via the parameter `Default`, as follows:

```
A := AssociativeArray(: Default := D);  // Null index universe
A := AssociativeArray(I: Default := D); // Index universe I
```

If a default value D has been specified, then whenever $A[x]$ is used but is not yet defined (i.e., x is not in the keys of A), then D is returned, instead of an error being raised. This allows constructions like the following, where the values of A will be sequences which are to be appended to dynamically:

```
A := AssociativeArray(: Default := []);
x := 3; y := 5;
Append(~A[x], y);
```

Here we can append an element to $A[x]$ even when x is not yet in the keys of A; in such a case, $A[x]$ is initially taken to be [] and so can be appended to without error.

6 Algebraic Geometry

6.1 Schemes

New Features:

– `DirectProduct` has been rewritten independently of the Toric Varieties for other ambients.

Changes and Removals:

– The grading of a `ProjectiveSpace` is now transferred to the coordinate ring even when that grading contains a zero entry. (V2.28-3)

– A crash in `PointSearch` for ambient spaces has been fixed. (V2.28-8)

6.2 Algebraic Curves

Bug Fixes:

– A hard coded limit in the preimage of the map from the function field into the algorithmic function field has been lifted. This limit interfered unnecessarily in the computation of a generating element for a principal divisor as tested by `IsPrincipal`.

7 Arithmetic Geometry

7.1 Hyperelliptic Curves and Jacobians

New Features:

- The intrinsic `RationalPointsGenus2` has been implemented by Michael Stoll, which attempts to compute the set of rational points on a hyperelliptic curve of genus 2. It uses a combination of techniques: search for (relatively small) points, test for everywhere local solubility, two-cover descent, degree 2 and 3 elliptic subcovers of rank zero, computation of generators of a finite-index subgroup of the Mordell-Weil group, Chabauty when the rank is at most 1, a check whether the curve has rational divisors of odd degree, if the rank is at least 2 and no points were found, a Mordell-Weil sieve computation.
- The printing of hyperelliptic curves has been improved in the case that the coefficient of the y-term is negative.

Changes and Removals:

- The intrinsic `ReduceModel` has been rewritten by Michael Stoll, and its optional parameters have changed. In particular, the parameters `Simple` and `Al` have been removed. The parameter `Smallest` and `Height` have been added to control the type of reduction.

7.2 Modular Symbols

New Features:

- The package for computing with Modular Symbols has been significantly sped up, particularly so that decomposition of spaces is often much faster than previously.

7.3 Algebraic Modular Forms (New)

Features:

- This package computes spaces of algebraic modular forms for definite orthogonal and unitary groups over an arbitrary totally real number field F, of weight which could be any finite dimesional representation of the group, and level $O(\Lambda)$ (or $U(\Lambda)$) where Λ is any integral positive definite $\mathbf{Z}_F$-lattice These spaces are created using `AlgebraicModularForms`, `OrthogonalModularForms` and `UnitaryModularForms`.
- Computations are done, via transversing the p-neighbours in the genus of a lattice, following an algorithm due to Greenberg and Voight; this is done internally and does not feature in the interface.
- The dimension of a space may be computed (which is a nontrivial computation, as dimension formulae are not available in general).
- Hecke operators can be computed, with respect to some fixed basis of the space, consisting of characteristic functions of the chosen representatives for the genus. (Currently there are no conventions regarding the choice of representatives.)
- The `Decomposition` of a space is available, as well as `HeckeEigenforms`, and systems of eigenvalues can be obtained for the modular form corresponding to each Hecke-irreducible component in the decomposition.

8 Arithmetic Fields (Global)

8.1 Algebraic Number Fields

New Features:

- Embeddings between number fields and orders of number fields can be retrieved using `HasEmbedding`. Whether an embedding can be computed between two fields or orders can be determined by `CanComputeEmbedding`.
- `Decomposition` of finite places of coefficient fields in extensions has been enabled. (V2.28-9)

Changes and Removals:

- The class group and unit group computation has undergone a thorough review. As a result, a new algorithm for the GRH-conditional residue of the Dedekind zeta function has been implemented. It can be called directly by `ResidueGRH` the error can be controled by `ResidueGRHbound`. Further, bugs in the saturation checks for class group and unit group were found and fixed. Using all implication of GRH, the proof levels of the class group have been reordered. Now, a GRH-conditional result is the lowest proof level. Without the specification of a proof level, the results are rigorous.
- `IsSubfield` and `IsIsomorphic` now respect existing embeddings. In addition `IsIsomorphic` no longer installs an embedding so there is no coercion resulting between the fields and the map returned needs to be used to move elements between the fields.
- Improvements to coercion between number fields and between orders of number fields have been made.
- `IsIsomorphic` now chooses coefficient embeddings that lead to subfield and isomorphism relationships over those that do not. (V2.28-3)
- Error checking has been improved for the `ext` constructor when the `Abs` parameter is `true`. (V2.28-4)

Bug Fixes:

- Expressing a field as a `RelativeField` of itself has been fixed. (V2.28-4)
- `Extends` now handles relative extensions properly. (V2.28-6)
- A bug affecting the integrality of an `IntegralBasis` computed using the Montes algorithm has been fixed. (V2.28-6)
- An "Ambiguous signature match" has been removed for `IntegralBasis`. (V2.28-6)
- Fixes have been made to recent improvements in MaximalOrder computations. (V2.28-9)

8.1.1 Quadratic Fields

Changes and Removals:

- `PicardGroup` is now computed using the implementation for orders of number fields rather than for quadratic forms. This allows preimages to be computed from the map returned.
- Parameters to `ClassGroup` have been reviewed and unused options removed.

8.2 Algebraic Function Fields

New Features:

- Embeddings between orders of function fields can be retrieved using `HasEmbedding`. Whether an embedding can be computed between two orders can be determined by `CanComputeEmbedding`.

Changes and Removals:

- `Decomposition` of primes in extensions of a polynomial ring or valuation ring now uses the order given as the first argument to decompose into rather than decomposing into a maximal order.
- Improvements have been made to `MaximalOrder` computations.
- `RationalFunction` for elements of non simple function fields now ensures the element has known coefficients rather than returning 0. (V2.28-9)
- Applying `Automorphisms` to elements with a product representation no longer computes coefficients of the element. (V2.28-9)
- Taking preimages of the identity automorphism has also been fixed. (V2.28-9)

Bug Fixes:

- A crash in working with elements of a quotient of an order of a function field has been fixed. (V2.28-8)

8.3 Galois Groups

New Features:

- Special support for polynomials of the shape $f(x) = g(x^d)$ has been added. To use is, the option `Deflation` has to be set in `GaloisGroup`.
- If a Galois group is computed twice with respect to different primes, an isomorphism of the results can be computed by `GaloisGroupConjugation`.

Changes and Removals:

- Precision handling has been improved in `SplittingField` so that the valuation of the discriminant can be accurately determined. (V2.28-9)
- User provided primes using the `Prime` parameter to `GaloisGroup` are now checked to be unramified. If a user given prime is ramified, another prime will be selected for the computation. (V2.28-9)
- Checking for intransitive groups has been added to `GaloisQuotient`. (V2.28-9)

9 Arithmetic Fields (Local)

9.1 p-adic Rings and their Extensions

Changes and Removals:

- A polynomial must now retain its degree when mapped to the residue class field in order to be inertial.
- Increased error checking of the map given as the second argument to `RelativeField` is now done. (V2.28-9)

Bug Fixes:

- The map returned from `AbsoluteTotallyRamifiedExtension` has been fixed. (V2.28-4)

9.2 Newton Polygons

New Features:

- The function `SlopesWithMultiplicities` has been added to display the slopes in various formats.

10 Basic Rings and Fields

10.1 Integer Ring

New Features:

- Primality testing and proving has been greatly improved as follows:
  1. For testing whether n is prime and $n < B$, where $B = 3317044064679887385961981$ (approx 3.3×10^{24}), the deterministic Sorenson/Webster method (with proof) is now used (using Miller-Rabin with the first 12 primes as bases).
  2. For testing whether n is prime and $n < 2^{64}$, a fast in-place Montgomery method is also now used, which is much faster than previously.
  3. A new version of the elliptic curve prime proof has been implemented. It can handle 1000 digit numbers in a few minutes, which is at least 10 times faster than previously. The primality of numbers with more than 5000 digits has already been proved with it. The function `PrimalityCertificate` returns the primality certificate generated by the elliptic curve prime proof. It can be verified and displayed with `CheckCertificate`. For compatibility, `OldCertificate` can be used to convert the new certificate to the old format.

11 Commutative Algebra

11.1 Univariate Polynomial Rings

New Features:

- Improvements have been made in the algorithms for the factorization of univariate polynomials over finite fields.

11.2 Multivariate Polynomial Rings

New Features:

- Some major improvements have been made to the algorithm for computing multivariate resultants. This benefits in particular the Trager algorithm for factoring multivariate polynomials over number fields.

11.3 Ideal Theory and Gröbner Bases

New Features:

- Major new improvements have been achieved for computing Gröbner bases of multivariate ideals where the coefficients lie in an algebraic number field. The new methods work particularly well for number fields of high degree (for which there had previously been no efficient methods in Magma) but are also faster than previously for fields of small degree.
- The critical algorithm for computing the primary decomposition of multivariate ideals has been improved in various ways, both for the zero-dimensional and positive-dimensional cases (involving quite different techniques and subalgorithms for each case). This algorithm is at the heart of decomposition of schemes and now allows the computation of some decompositions to be feasible which were not previously.

12 Groups

12.1 Reductive Groups (New)

Features:

- This package provides interface for reductive groups such as orthogonal, symplectic and unitary groups over a number field. Currently supports groups that are split extensions of their identity component (but not necessarily connected), and such that each of the simple factors in their almost simple factorization is either orthogonal or unitary.
- Given a symplectic, orthogonal or unitary vector space over a numebr field, one can form the corresponding `SymplecticGroup`, `OrthogonalGroup`, or `UnitaryGroup`. One can also use `ReductiveGroup` to create a reductive group from a group of Lie type specifying its component of the identity, and a finite group specifying its component group.
- One can also create elements in these groups, and perform simple arithmetic with such elements.

13 Language and System Features

13.1 Language

New Features:

– A comma-separated list with N expressions is now allowed for the right-hand-side (RHS) of an assignment statement, when the number of lvalues on the left to be assigned equals N. The expressions on the RHS are all evaluated first before any assignment on the left. For example, the following statement can now be used to swap variables a and b:

```
a, b := b, a;
```

Note that for each expression on the RHS which has multiple return values, all those values are discarded except for the first. For example, the following statement will set a to 1 and b to 7 (only the first return value is used for each call of Quotrem):

```
a, b := Quotrem(13, 10), Quotrem(75, 10);
```

– A major improvement has been done in the language interpreter so that reference arguments with indexing in procedure calls are handled much more efficiently, and undesired temporary cloning of objects is now avoided (e.g., in procedure calls involving the tilde operator before indexed references such as `S[i]`, and operators like `+:=` with indexed references on the LHS).

14 Lattices and Quadratic Forms

14.1 Lattices

New Features:

– Lattices over number nields now support Hermitian lattices, by specifying the optional parameter `Involution` in the constructor `NumberFieldLattice`.

– `IsIsometric` and `AutomorphismGroup` now support the optional parameter `Proper`, in order to considered only proper (orientation-preserving) isometries.

– `Genus` now supports displaying ang generating genus symbols, with `Genera` for generating all genus symbols of given signature and determinant.

– `GenusRepresentatives` of lattices nowuse the mass formula for early termination.

Changes and Removals:

– `WittInvariant` has been modified to be consistent with number field lattices and with the literature. It is no longer equivalent to the `HasseInvariant`.

Bug Fixes:

– A crash in `IsGLZConjugate` involving short vector enumeration has been fixed.

– Fixed a bug in `IsIsometric` for non-free lattices.

15 Lie Theory

15.1 Representation Theory

Bug Fixes:

– An internal check that a root datum is (weakly) simply connected has been corrected. This sometimes caused a runtime error in `LieRepresentationDecomposition`. This bug was reported by Robert Zeier.

16 Linear Algebra and Module Theory

16.1 Matrices

New Features:

– Major new improvements have been achieved for computing with matrices where the coefficients lie in an algebraic number field (this includes algorithms for computing nullspaces, echelon forms and determinants). The new methods work particularly well for number fields of high degree (for which there had previously been no efficient methods in Magma) but are also faster than previously for fields of small degree.

16.2 Sparse Matrices

New Features:

– The functions `MinimalPolynomial`, `CharacteristicPolynomial`, `Eigenvalues`, `Eigenspace`, etc. have been added for sparse matrices.

17 Linear Associative Algebras

17.1 Matrix Algebras

Changes:

– A matrix M is diagonalisable if its Jordan form is diagonal; i.e., there is an invertible matrix T such that TMT^{-1} is diagonal. Alternatively a symmetric matrix is diagonalisable if it is the Gram matrix of an orthogonal basis; i.e., there is a matrix T such that TMT^{tr} is diagonal.

To distinguish between these two types of diagonalisation, the intrinsic `Diagonalization` that applies to symmetric matrices has been renamed `Orthogonalization`.

The intrinsic which diagonalises a matrix in the sense of finding a diagonal Jordan form retains the name `Diagonalization`. Furthermore, the names `Diagonalisation` and `Diagonalization` are now synonyms.

There is a new intrinsic `IsDiagonalisable` (and `IsDiagonalizable` is a synonym). The default behaviour checks whether a matrix is diagonalisable over its field of definition. If the parameter

`ExtendField` is `true`, diagonalisation is carried out over the splitting field of its characteristic polynomial.

A new intrinsic `IsEtale` checks whether an algebra is étale; i.e., diagonalisable over an extension field. If so, a matrix L is returned such that for all elements M of the algebra, LML^{-1} is diagonal.

The intrinsic `CommonEigenspaces` returns common eigenvalues and eigenspaces of a sequence Q of commuting matrices. This intrinsic now has parameters `U` and `Check`. If U is assigned, only subspaces of U are considered. The default value for U is the base module of the universe of Q. If `Check` is `true` (the default value), the matrices in Q are checked for commutativity.

17.2 Étale Algebras

A package for étale algebras developed by Stefano Marseglia has been added. It covers étale algebras that are products of absolute number fields and supports orders and ideals.

17.3 Finitely Presented Associative Algebras

New Features:

- The computation of noncommutative Gröbner bases for Finitely Presented Associative Algebras has been greatly improved:

  1. The *useless pair elimination* method of B. Keller (the noncommutative analogue to the Gebauer-Möller criterion) has been implemented. This means that in general, many less critical pairs are needed than previously, so the size of the matrices in the noncommutative F_4 algorithm are much smaller.
  2. The *Aho-Corasick substring search algorithm* has been implemented for words in FP algebras (for a dictionary D of words and given word S, this finds all words in D which are subwords of S). This allows a very fast search for divisors in D of a given noncommutative monomial S, where D are the leading monomials of a partial Gröbner basis.

  As a result of both points above, the symbolic reduction phase (which searches for reductors) and the critical pair phase (which does a similar search in the elimination search) have both been very greatly sped up for many kinds of large examples.

18 Representation Theory

18.1 Character Theory

New Features:

- The Dixon-Schneider algorithm for computing the character table of a finite group has been greatly sped up.
- The default algorithm for computing the character table of a finite group now uses the Dixon-Schneider algorithm for groups with small conjugacy classes.
- A new fast algorithm for computing the character table of an abelian group has been developed.
- Addition of characters (such as returned by `GaloisOrbit`) has been greatly sped up.

18.2 Group Representations (New)

Features:

- This package computes group representations for groups that are not necessarily finitely generated, with focus on reductive groups.
- Standard constructions such as `StandardRepresentation`, `TrivialRepresentation` and `AlternatingRepresentation` are available.
- For a reductive group, one can also construct a `HighestWeightRepresentation`, and for orthogonal groups one can construct thr `SpinorNormRepresentation`, as well as characters induced from the action on the radical via `RadicalSignCharacter`.