Summary of New Features in Magma V2.24

August 2018

1 Introduction

This document provides a terse summary of the new features released as part of Magma versions V2.24 (August 2018).

A small number of new features were exported in patch releases prior to the main release of V2.24 in August 2018 and these are also listed here for completeness. Only significant bugfixes are noted here – for a more complete list of bugfixes the reader should consult the patch release change log for V2.23-x.

Recent releases of Magma were: V2.23 (July 2017), V2.22 (May 2016), V2.21 (December 2014), V2.20 (December 2013), V2.19 (December 2012), V2.18 (December 2011), V2.17 (December 2010), V2.16 (November 2009), V2.15 (December 2008), V2.14 (October 2007).

2 Highlights

Arithmetic Geometry

- *Riemann Surfaces*
  - A package for computing with Riemann surfaces defined by an affine equation $f(x, y) = 0$ is currently being implemented by Christian Neurohr. The algorithms are described in his 2018 PhD thesis at Universität Oldenburg. This package will not be part of the V2.24 release but instead will be included in an upcoming patch release.

    The main focus is on approximating period matrices and the Abel-Jacobi map of divisors using numerical integration up to a prescribed precision. An object of Riemann surface type is defined by an irreducible polynomial $f \in K[x, y]$ and P, where K is $\mathbf{Q}$ or a number field, and P is a complex embedding or in the superelliptic case by $f = y^m - h(x)$ with $h \in C[x]$ squarefree and $m > 1$. Working with a precision of hundred decimal digits is practical in the general case and several thousand digits in the superelliptic case.

- *Hypergeometric Motives*
  - A new package for computing associated schemes in product projective spaces has been contributed by Bartosz Nasrecki. In typical cases this allows a reduction in the dimension of the resulting variety, via detection of zerosum-subsets of the list of gamma parameters. Although finding such zerosum-subsets is an NP-hard problem in general, in our specific cases we can exploit various symmetries (working in fact with zerosum-submultisets) to realize a reasonably fast solution. For instance, it takes about one minute to compute the associated schemes for all the 8444 examples of degree 8 hypergeometric data. The typical reduction of the dimension is 4, but can be as much as 18.
  - The `HypergeometricTraceK` intrinsic, applicable for non-Galois data and/or parameter specializations at non-rationals, has been improved and documented.
  - A number of identifications of weight 0 motives as Artin representations has been contributed by Bartosz Nasrecki. This includes all examples up through degree 5.

Arithmetic Fields

- *Galois Groups*
  - Code has been added to compute the geometric Galois group of a polynomial over $\mathbf{Q}(t)$.
  - A new intrinsic provides information about the set of exceptions to Hilbert's Irreducibility Theorem in the case of polynomials having coefficients that are rational functions over the rational field.

Basic Rings and Fields

- *Polynomial Rings*
  - New parameter `DegreeLimit:=D` added to `Factorisation` for univariate polynomials so that only irreducible factors with degree at most the given bound D are returned (this can lead to large speedups for certain base rings).
  - The von zur Gathen/Kaltofen/Shoup algorithm for factoring polynomials in $K[x]$, where K is a finite field, has been improved by better use of the Brent-Kung algorithm. The speedup is up to a factor of 2 when factoring polynomials over medium-sized finite fields.
  - The general factorisation algorithm for univariate polynomials over $\mathbf{Z}$ or $\mathbf{Q}$ has been sped up for several types of input, by improvement of the general strategy (especially for recursively-defined rings with several levels).

- The resultant algorithm has been greatly improved for polynomials defined over residue rings with zero divisors and similar generic commutative rings (by use of the Berkowitz-based determinant algorithm).
- The resultant algorithm has been sped up in the case where the coefficients lie in a subring.

Coding Theory

Major improvements have been made to the fundamental minimum weight and weight distribution algorithms.

- *Linear Codes over Finite Fields*
  - For linear codes over GF(2), the minimum weight algorithm has been greatly sped up. For codes of length up to 256 and relatively large dimension, the speedup can be a factor of about 15, while for codes of much longer length, the speedup is typically 5 to 10. The related functions for finding the words of minimum weight or words of bounded weight, etc. have similar speedups. As an example, the minimum weight of the [193, 97] quadratic residue code over GF(2) is proven to be 27 in about 2.7 days on a single 2.3GHz 36-core Intel E5-2699 v3 CPU (this is 15.5 times faster than Magma V2.23).
  - For linear codes over $\mathbf{Z}_4$ or GF(q) ($q > 2$), the minimum weight algorithm has been sped up by a factor of 1.5 to 2, typically. There are similar speedups for the enumeration of the words of minimum weight, etc.
  - For linear codes over GF(2), the algorithm for determining the weight distribution has been sped up, typically by a factor of about 5. Speedups for determining the weight distribution over $\mathbf{Z}_4$ and other finite fields have also been achieved.
  - Multiple threads are now supported for the minimum weight algorithm (and all the above related algorithms) for linear codes defined over $\mathbf{Z}_4$ and GF(q) ($q > 2$). Only the GF(2) case had been supported previously. The times which are printed as the algorithm progresses (with the verbose flag turned on) are now real (wall clock) times.
  - Multiple threads are now supported for computing weight distribution of linear codes for all base rings.
  - The MacWilliams Transform algorithm has been considerably sped up.

Commutative Algebra

- *Gröbner Bases*
  - The dense Gröbner basis linear algebra phase has been improved, both for the GF(2) and GF(p) cases (medium prime p). The general strategy has been improved too.
  - The FGLM GB monomial order change algorithm has been non-trivially improved for ideals defined over medium prime finite fields.
  - Better support has been developed for fast GB computations for ideals defined over number fields defined in terms of multiple relative extensions.
- *Ideal Arithmetic*
  - Several major improvements have been made to the primary decomposition algorithm.
  - Major improvements have been made to the algorithms for computing intersections of ideals and colon ideals of the form $I : J$.

Group Theory

- *Finitely-Presented Groups*
  - An intrinsic has been added which given an fp-group G and an epimorphism from G onto a permutation group, applies the Holt-Plesken criteria in an attempt to prove that G is infinite.
  - Improvements have been made to the package that attempts to prove that a given fp-group is infinite by searching for suitable epimorphisms of G onto permutation groups and then applying the Holt-Plesken criteria (intrinsic `IsInfiniteFPGroup`).
  - In V2.23 code developed by Derek Holt for attempting to show that a finitely-presented group is word-hyperbolic was released. The method is based on a forthcoming paper by Holt, Linton, Neunhöffer, Parker, Pfeiffer and Roney-Dougal, provisionally entitled *A new approach to proving hyperbolicity.* As the test is somewhat difficult to apply, much more extensive Handbook documentation, including a number of new examples, is provided in this release.
- *Classical Groups*
  - Fast computation of the conjugacy classes in finite unitary groups based on the use of class invariants is now provided. The new code computes the conjugacy classes for subgroups of the general unitary group that contain the special unitary group by first enumerating class invariants and then finding a representative matrix for each invariant. There is no restriction on the characteristic of the field.

    The finite classical groups for which this method is now implemented are:
    * Groups containing the special linear group;
    * Subgroups of the conformal symplectic group that contain the symplectic group in *odd* characteristic;
    * The conformal unitary group;
    * Subgroups of the general unitary group that contain the special unitary group;
    * The general orthogonal groups in *odd* characteristic;
    * The conformal orthogonal groups in *odd* characteristic.

Linear Algebra and Module Theory

- *Linear Algebra over Finite Fields*
  - The base algorithm for matrix multiplication in moderately small dimension has been greatly improved for matrices over GF(q) for $q = 2, 3, 4, 5, 7$. For example, matrix inversion over GF(2) in dimension up to 128 is roughly 3 to 4 times faster on a typical Intel Core CPU.
  - The base algorithm for matrix echelonisation has been greatly improved in general for matrices of small to medium size with entries in GF(q) for $q = 2, 3, 4, 5, 7$. For example, matrix inversion over GF(2) in dimension up to 1000 is roughly 3 to 4 times faster on a typical Intel Core CPU.
  - Matrix multiplication over GF(p) for $p = 3, 5, 7$ has been significantly improved (for all matrix sizes) in the AVX version.
- *Linear Algebra over General Rings*
  - The recursive echelon algorithm has been improved in the case that a transformation matrix is required. This also yields improvements to many algorithms which depend on this, including the computation of the inverse of a matrix.
  - The Hermite normal form algorithm for matrices over the integer ring has been improved for some types of matrix where there are many non-trivial repeated elementary divisors.
  - New interpolation-based algorithms have been implemented for computing the determinant and characteristic or minimal polynomial of matrices over a ring R, where R is a integer or polynomial residue ring, or a polynomial ring over such a ring. These are much faster than the previous generic algorithms when applied to matrices of non-trivial sizes.
  - The worst-case exponential-time algorithm for computing the characteristic polynomial of a matrix defined over a general commutative ring has been replaced with the Berkowitz algorithm, which takes $O(n^4)$ ring operations for an $n \times n$ matrix. In particular, this speeds up the computation of the characteristic polynomial of a matrix defined over a ring with zero divisors.
  - A new generic polynomial-time determinant algorithm has been implemented, based on the Berkowitz algorithm above. This algorithm is applicable to rings with zero divisors for which there is no standard Gaussian elimination-based algorithm available, and is much faster in higher dimensions than the generic exponential algorithm which was previously used.
  - New functions `RowWeights` and `ColumnWeights` have been provided for dense matrices.

Representation Theory

- *KG-Modules*

  - A package implemented by Peter Brooksbank, based on a new algorithm due to him and Eugene Luks for testing two A-modules for isomorphism, has been installed. The algorithm is superior to the previous algorithm in the case of difficult reducible A-modules.
  - Isomorphism of difficult reducible KG-modules is now decided using the Brooksbank-Luks algorithm (see above) resulting in much reduced runtimes.
  - Intrinsics are now provided for computing vertices and sources for KG-modules, where K is a finite field.
  - An intrinsic for computing the injective hull of a KG-module over a finite field has been installed. This algorithm was suggested by Dave Benson and implemented by Derek Holt.
  - It is now possible to partition, into blocks, the set of projective indecomposable KG-modules, where K a finite field.
  - The composition factors of the modules obtained by taking the quotients of each pair of adjacent terms of the socle series for a KG-module, K a finite field, can now be displayed.

System

- *Magma Startup*

  - The startup time for Magma has been greatly reduced; it is now typically only 0.02 seconds on a standard Linux installation. This allows many very short successive Magma jobs (called from perl/shell scripts, etc.) to be run much more efficiently.

3 Algebraic Geometry

3.1 Schemes

Changes and Removals:

- Blowup has been corrected so that extra linear components, which occasionally appeared before and are not mathematically part of the actual blow-up, no longer appear. In relation to this, a new parameter Xirred has been added to the intrinsic. It allows a simplification of the saturation process that removes the extra components.
- The Blowup function has been sped up.
- The computation of images of subschemes under scheme maps now makes use of alternative defining polynomials for the map. These were previously only used for images of points. Now the image of a subscheme that has components which lie in the base scheme of the primary defining polynomials but are covered by a set of alternative defining polynomials correctly contains the images of these components.

Bug Fixes:

- A crash in the application of the PolynomialMap of a Linear system a sequence of vectors has been fixed. (V2.23-8)
- A bug in the Rees Ideal computational part of Blowup, which was causing some runtime errors, has been fixed. (V2.23-10)
- An internal problem with ambients that are the product of an affine and ordinary projective space has been fixed. This was causing a number of errors and crashes when working with subschemes of these ambients. (V2.23-10)

3.2 Algebraic Curves

Changes:

- The Flexes and InflectionPoints intrinsics now work for affine as well as projective plane curves. (V2.23-10)

Bug Fixes:

- A check has been added to ensure that a curve has a function field before a divisor of the curve can be constructed. (V2.23-10)
- A bug in Degree for non-constant maps between curves which caused errors when the domain of the map was not of Crv type has now been fixed. (V2.23-10)
- A bug has been fixed in Automorphisms and AutomorphismGroup which was producing internally-inconsistent automorphisms for curves or function fields over a finite field defined as a non-standard extension of the ground field GF(p). This was causing runtime errors in a number of examples.

4 Arithmetic Geometry

4.1 Riemann Surfaces

A package for computing with Riemann surfaces defined by an affine equation $f(x, y) = 0$ that is currently being implemented by Christian Neurohr will be distributed in an upcoming patch release. The algorithms are described in his 2018 PhD thesis at Universität Oldenburg:
`https://oatd.org/oatd/record?record=oai%5C%3Aoops.uni-oldenburg.de%5C%3A3607`.

Features:

- The package introduces a Riemann surface type `RieSrf` together with types for points `RieSrfPt` and divisors `DivRieSrfElt`.
- An object of Riemann surface type is defined by in one of two ways:
  - As a general curve defined by an irreducible polynomial $f \in K[x, y]$ and a mapping P where K is $\mathbf{Q}$ or a number field, and P is a complex embedding;
  - As a superelliptic curve defined by $f = y^m - h(x)$ with $h \in C[x]$ squarefree and $m > 1$.
- Working with a precision of several hundred decimal digits is practical in the general case and several thousand digits in the superelliptic case.
- Period matrices and the Abel-Jacobi map of superelliptic curves will be much faster than the general ones.
- Several different algorithms for numerical integration are available, namely: Double-exponential, Gauss-Legendre or Clenshaw-Curtis quadrature in the general case and double-exponential or Gauss-Jacobi quadrature in the superelliptic case.
- The most important intrinsics for Riemann surfaces are: `RiemannSurface`, `SmallPeriodMatrix`, `BigPeriodMatrix`, `FundamentalGroup`, `AnalyticContinuation`, `MonodromyRepresentation`, `HomologyBasis`, and `AbelJacobi`.
- Intrinsics that perform numerical integration are: `GaussJacobiIntegrationPoints`, `GaussLegendreIntegrationPoints`, `ClenshawCurtisIntegrationPoints`, `DiscreteFourierTransform`, and `TanhSinhIntegrationPoints`.

4.2 Rational Curves and Conics

Bug Fixes:

- A problem with reducible input was fixed. (V2.23-6)

4.3 Elliptic Curves

4.3.1 Elliptic Curves over Number Fields

Bug Fixes:

- The `MordellWeilGroup` machinery (including also `Saturation` and `MordellWeilShaInformation`) should now work for non-integral models (over number fields) and also over non-absolute extensions. (V2.23-10)

4.3.2 Elliptic Curves over Finite Fields

Bug Fixes:

- A bug with the SEA point-counting algorithm over finite fields defined by relative extensions was fixed. (V2.23-2)

4.4 Hyperelliptic Curves and Jacobians

New Features:

- The function `TwoCoverDescent` has been improved by M. Stoll. Also, a new parameter Points has been added, which can be set to a set of rational points on the curve, to be used for an early exit when the given points exhaust the upper bound for the Selmer set computed so far. (V2.23-5)

Bug Fixes:

- The 3rd return value of the function AutomorphismGroup has been fixed for hyperelliptic curves. (V2.23-5)
- An incorrect answer for `ToAnalyticJacobian` at Weierstrass points has been corrected. (V2.23-6)
- A possible crash when trying to make sequences of tuples containing points from incompatible hyperelliptic curves has been fixed. (V2.23-8)
- A problem with heights (derived from Kummer surface computations) for torsion points on the Jacobians was fixed. Also, the height is now returned as zero to the user (rather than default) precision. (V2.23-9)
- A fix to `LPolynomial` and `ZetaFunction` for hyperelliptic curves which don't have type `CrvHyp` has been made. (V2.23-10)

4.5 Hypergeometric Motives

New Features:

- The `HypergeometricTraceK` intrinsic, applicable for non-Galois data and/or parameter specializations at non-rationals, has been improved and documented. (V2.23-2)
- The intrinsic MonodromyGroup was added. (V2.23-10)

Bug Fixes:

- A problem with a non-integral wild Euler factor was fixed. (V2.23-6)
- Identifying a hypergeometric motive which is a hyperelliptic curve now automatically sets the LocalData parameter to “Ogg” in its L-series. (V2.23-7)
- The `EulerFactor` intrinsic now immediately raises a size error when the prime powers would be too large, rather than incrementally computing until an error occurs. (V2.23-10)
- Some problems with unbalanced hypergeometric data (used in computing guesses for wild primes) were fixed. (V2.23-10)
- Minor problems with “common factors” were fixed in some intrinsics. (V2.23-10)
- Some discrepancies with `ComplexEvaluation`, and its underlying usage in `Grossencharacter` were fixed.

4.6 L-Series

Bug Fixes:

- Taking an `EulerFactor` of the trivial (unital) L-series no longer gives an error. (V2.23-7)
- A fix to `LPolynomial` and `ZetaFunction` for hyperelliptic curves which don’t have type CrvHyp has been made. (V2.23-10)

4.7 Modular Forms

Bug Fixes:

- A crash in `DualHeckeOperator` has been fixed. (V2.23-4)

4.8 Arithmetic Fuchsian Groups

Bug Fixes:

- A bug with `FundamentalDomain` when an isometric circle meets the real axis at a right angle was fixed. (V2.23-7)

5 Arithmetic Fields (Global)

5.1 Dirichlet and Hecke Characters

Bug Fixes:

- Some problems with coercion of the trivial character were fixed, and the code now caches ideals to ensure compatability. (V2.23-5)
- The evaluation functions previously coerced 1 and -1 to the Integers; this has been removed in the case that the evaluations are not elements of a cyclotomic field in the first place, thus making the usage of SetTargetRing (e.g. for a finite field) give more uniform answers. (V2.23-7)

5.2 Algebraic Number Fields

New Features:

- Several attributes for orders and ideals have been added. These allow the retrieval of known information without computing such information if it is not known, which can be done using the relevant intrinsics. These attributes are `MaximalOrder` and `ArithmeticField` for orders and `BasisMatrix`, `IsPrime`, `PartialFactorization`, `InertiaDegree` and `RamificationIndex` for ideals.

Changes and Removals:

- More information is passed on where possible when an ideal is constructed from another ideal.
- The `OptimizedRepresentation` of a `FldRat` is now available.

Bug Fixes:

- The `PowerRelation` intrinsic now ensures its output is monic, and there is some attempt to remove superfluous factors in it too. (V2.23-2)
- The GenusField intrinsic has been corrected in some cases where the original abelian extension is not absolutely abelian. (V2.23-6)
- Computing an order from a sequence of elements in a number field represented as a tower of extensions has been fixed. (V2.23-6)

5.2.1 Cyclotomic Fields

Bug Fixes:

- A deletion problem with cyclotomic fields has been fixed. This was seen during computations with character tables and class groups. (V2.23-5)

5.3 Characters and Artin Representations

New Features:

- A new intrinsic `RationalDecomposition` of an Artin representation has been added, its usage parallelling that of `RationalCharacterTable`. In particular, the resulting characters are rational-valued (not rationally represented).

Bug Fixes:

- An incorrect answer (due to too low of precision) for various Frobenius computations (particularly at 2) has been fixed. (V2.23-7)
- The `Minimize` intrinsic should no longer return `false` when the vararg `Optimize` is `false` and the result did not change.

5.4 Algebraic Function Fields

New Features:

- The p-adic `Development` of an algebraic function can now be computed, giving an algebraic function as a combination of powers of a prime.
- Several attributes for orders and ideals have been added. These allow the retrieval of known information without computing such information if it is not known, which can be done using the relevant intrinsics. These attributes are `MaximalOrder` and `ArithmeticField` for orders and `BasisMatrix`, `IsPrime`, `PartialFactorization`, `InertiaDegree` and `RamificationIndex` for ideals.

Changes and Removals:

- Improvements have been made to the quotients of orders of function fields. (V2.23-3)
- The product of an ideal in 2-element representation with itself is now returned in 2-element representation.
- An ideal constructed using `ideal< | >` where the right hand side contains a sequence of 1 or 2 elements is now handled the same as if the elements were not in a sequence, that is, the ideal will be constructed as principal or in 2 element representation rather than a basis be computed immeadiately.
- More information is passed on where possible when an ideal is constructed from another ideal.

Bug Fixes:

- A fix has been made to `ClassNumber` of a function fields to avoid division by zero. (V2.23-11)
- The degree of the different divisor has been fixed for fields having more than 2 relative extensions in their representation. (V2.23-11)

5.5 Galois Groups

New Features:

- The intrinsic `GeometricGaloisGroup` computes the geometric Galois group of a polynomial over $\mathbf{Q}(t)$.
- The intrinsic `HilbertIrreducibilityCurves` provides information about the set of exceptions to Hilbert's Irreducibility Theorem for polynomials with coefficients that are rational functions over the rational field.

Bug Fixes:

- A precision check in the computation of Galois groups of reducible polynomials over $\mathbf{Q}(t)$ has been fixed. (V2.23-2)
- The computation of subfields and Galois groups was improved by a refined cycle type analysis. (V2.23-8)
- An infinite loop has been broken in the computation of a prime, which does not divide the discriminant of the defining polynomial of any subfield, to use in the computation of a Galois group of a polynomial over a function field. (V2.23-8)

6 Arithmetic Fields (Local)

6.1 p-adic Rings and their Extensions

Bug Fixes:

- A bug with coercion of rational zeros into a p-adic field has been fixed. Such an p-adic number could be created, but would not be correctly be recognized as zero in comparisons. (V2.23-2)
- A fix has been made to the `Factorization` of polynomials over local rings. (V2.23-4)
- Zeroth powers of zero are now 1 instead of an imprecise zero. (V2.23-6)
- A bug in testing whether a polynomial is inertial has been fixed. (V2.23-8)
- The `Eltseq` of infinite valuation zeros has been fixed. (V2.23-9)
- The `NumberOfExtensions` intrinsic did not correctly adjust for non-absolute input. (V2.23-9)
- Compatibility checking has been improved in `NormGroup`. (V2.23-10)
- A finite precision is now checked for in `UnitGroup`. (V2.23-10)
- Error checking of input to `Exp` has been improved. (V2.23-11)
- For elements in a tower of extensions which includes a ramified extension `Exp` has been corrected to no longer return a zero as the result of intermediate overflow.

6.2 Series Rings

Bug Fixes:

- An error from the `Roots` computation for polynomials over infinite precision series rings about not enough precision available has been resolved by the fixing of a bug in the computation of GCDs of polynomials over series rings. (V2.23-3)
- A fix has been made to the `Roots` computation for polynomials over series rings whose degree is the characteristic of the coefficient ring. (V2.23-4)
- The computation of `GCD`s of polynomials over series rings one of whose degrees is greater than the precision of the ring has been fixed.

7 Basic Rings and Fields

7.1 Integer Ring

Bug Fixes:

– The upper limit on `PrimesUpTo` has been changed so that a crash no longer ensues. (V2.23-5)

7.2 Real and Complex Fields

Bug Fixes:

– A problem with epsilonic real/imaginary parts of `Roots` was fixed, so that `IsReal` should now work as previously. (V2.23-6)

7.3 Polynomial Rings

New Features:

– New parameter `DegreeLimit:=D` added to `Factorisation` for univariate polynomials so that only irreducible factors with degree at most the given bound D are returned (this can lead to large speedups for certain base rings).

– The von zur Gathen/Kaltofen/Shoup algorithm for factoring polynomials in $K[x]$, where K is a finite field, has been improved by better use of the Brent-Kung algorithm. The speedup is up to a factor of 2 when factoring polynomials over medium-sized finite fields.

– The general factorisation algorithm for univariate polynomials over $\mathbf{Z}$ or $\mathbf{Q}$ has been sped up for several types of input, by improvement of the general strategy (especially for recursively-defined rings with several levels).

– The resultant algorithm has been greatly improved for polynomials defined over residue rings with zero divisors and similar generic commutative rings (by use of the Berkowitz-based determinant algorithm).

– The resultant algorithm has been sped up in the case where the coefficients lie in a subring.

Bug Fixes:

– A missing error check has been included to ensure that divisors are non zero when using `div`. (V2.23-8)

8 Coding Theory

8.1 Linear Codes: Minimum Weight and Weight Distribution

Some major improvements have been made to the fundamental minimum word and weight distribution algorithms.

New Features:

- For linear codes over GF(2), the minimum weight algorithm has been greatly sped up. For codes of length up to 256 and relatively large dimension, the speedup can be a factor of about 15, while for codes of much longer length, the speedup is typically 5 to 10. The related functions for finding the words of minimum weight or words of bounded weight, etc. have similar speedups. As an example, the minimum weight of the [193, 97] quadratic residue code over GF(2) is proven to be 27 in about 2.7 days on a single 2.3GHz 36-core Intel E5-2699 v3 CPU (this is 15.5 times faster than Magma V2.23).
- For linear codes over $\mathbf{Z}_4$ or GF(q) ($q > 2$), the minimum weight algorithm has been sped up by a factor of 1.5 to 2, typically. There are similar speedups for the enumeration of the words of minimum weight, etc.
- For linear codes over GF(2), the algorithm for determining the weight distribution has been sped up, typically by a factor of about 5. Speedups for determining the weight distribution over $\mathbf{Z}_4$ and other finite fields have also been achieved.
- Multiple threads are now supported for the minimum weight algorithm (and all the above related algorithms) for linear codes defined over $\mathbf{Z}_4$ and GF(q) ($q > 2$). Only the GF(2) case had been supported previously.

  As before, the parameter `Nthreads := n` can be used to specify n threads for the individual call, or the procedure statement `SetNthreads(n);` can be used to specify n threads for all uses of the above algorithms.
- Multiple threads are now supported for computing weight distribution of linear codes for all base rings (via `Nthreads` or `SetNthreads` as above).
- The MacWilliams Transform algorithm has been considerably sped up.

8.2 Linear Codes over Finite Rings

This release contains an update of the package for linear codes over $\mathbf{Z}_4$, which is being developed by Merce Villanueva and the Combinatoric, Coding and Security Group (CCSG) at the Universitat Autònoma de Barcelona. It includes the following new intrinsics:-

New Features: Barcelona:

- `StandardFormDual`: The dual of a permutation-equivalent code S in standard form,
- `MinRowsParityCheckMatrix`: A parity check matrix for the code C over $\mathbf{Z}_4$ of length n and type $2^\gamma 4^\delta$, with the minimum number of rows.
- `DualZ4`: The dual D of the code C over $\mathbf{Z}_4$ of length n.
- `PermutationGroup`: The permutation group G of the linear code C of length n over the ring R, where G is the group of all permutation-action permutations which preserve the code.

– `PermutationGroupGrayMapImage`: Given a code C over $\mathbf{Z}_4$ of length n, return the permutation group G_{bin} of $C_{bin} = \Phi(C)$, where G_{bin} is the group of all permutation-action permutations which preserve the binary code C_{bin} of length $2n$ and Φ is the Gray map. Thus, only permutation of coordinates is allowed, and the degree of G_{bin} is always $2n$.

New Features: Sydney:

– For linear codes over $\mathbf{Z}_4$, the single processor minimum weight algorithm has been sped up by a factor of 1.5 to 2, typically. There are similar speedups for the enumeration of the words of minimum weight etc. Speedups have also been achieved for determining the weight distribution.

– Multiple threads are now supported for the minimum weight algorithm, the word enumeration algorithm and the weight distribution algorithm for linear codes defined over Z_4.

9 Commutative Algebra

9.1 Gröbner Bases

New Features:

- The dense Gröbner basis (GB) linear algebra phase has been improved, both for the GF(2) and GF(p) cases (medium prime p). The general strategy has been improved too.
- The FGLM GB monomial order change algorithm has been non-trivially improved for ideals defined over medium prime finite fields.
- The FGLM GB monomial order change algorithm has been very greatly improved for ideals defined over rational function fields, by use of fraction-free methods in the linear algebra operations.
- Better support has been developed for fast GB computations for ideals defined over number fields with multiple relative extensions.

Bug Fixes:

- A crash in the F_4 algorithm which arose when calling the function `ProbableRadicalDecomposition` has been fixed. (V2.23-12)
- A hang in the Gröbner Walk algorithm with ideals containing dense polynomials has been fixed. (V2.23-12)
- A rare incorrect result in the F_4 algorithm for ideals defined over medium characteristic finite fields has been fixed. (V2.23-12)
- A rare crash in the F_4 algorithm for ideals over number fields has been fixed.

9.2 Ideal Theory

New Features:

- Several major improvements have been made to the primary decomposition algorithm. These include:
  1. A major speedup in the algorithm for the decomposition of zero-dimensional ideals over rational function fields.
  2. A new heuristic which is applicable to radical multivariate polynomial ideals of positive dimension. Several decompositions of radical ideals which previously took a large amount of time are now done very quickly. Such ideals occur often, particularly when working with plane curves.
- Major improvements have been made to the algorithms for computing intersections of ideals and colon ideals of the form $I : J$.

9.3 Modules over Multivariate Polynomial Rings

New Features:

- The `Regularity` intrinsic that computes the Castelnuevo-Mumford regularity of a graded module has been updated to handle the dimension zero (Artinian) case much more efficiently, using only the Hilbert series rather than the full Betti table.

10 Groups

10.1 Finite Groups

Changes:

- An update of algorithms for computing subgroups of finite matrix and permutation groups has been undertaken to reduce time taken and increase the range of group sizes the algorithms will work on.

10.2 Classical Groups

New Features:

- Fast computation of the conjugacy classes in finite unitary groups is now provided. The `Classes` (equivalently `ConjugacyClasses`) function uses the fast methods whenever Magma detects that the group in question is a standard classical group of one of the following types:-
  - Groups containing the special linear group;
  - Subgroups of the conformal symplectic group that contain the symplectic group in *odd* characteristic;
  - The conformal unitary group;
  - Subgroups of the general unitary group that contain the special unitary group;
  - The general orthogonal groups in *odd* characteristic;
  - The conformal orthogonal groups in *odd* characteristic.
- The class invariants and representative matrices are available via functions of the form `ConjugacyInvariantX`, `RepresentativeMatrixX` and so on, where X is the name of the group. For example, functions installed to support the unitary groups include:
  - `ConjugacyInvariantGU`(g), where g is a unitary matrix;
  - `RepresentativeMatrixGU(inv)`, where `inv` is a unitary invariant;
  - `ClassInvariantsGU`(d, q), where d is the degree and q is the field size;
  - In addition there are versions of these functions for the *extended* special unitary groups; i.e., subgroups of the the general unitary group that contain the special unitary group.

10.3 Finitely Presented Groups

New Features:

- The new intrinsic `HasPositiveH1Dimension`, given a finitely-presented group G and an epimorphism from G onto a permutation group, applies the Holt-Plesken criteria in an attempt to prove that G is infinite.
- The performance of intrinsic `IsInfiniteFPGroup` which attemps to prove that a given finitely-presented group is infinite has been improved. Users are reminded that for intrinsics `HasPositiveH1Dimension` and `IsInfiniteFPGroup`, the optional Magma database of rational representations should be installed in order to get the best performance.

– The intrinsic `RSym`, introduced in V2.23 and which attempts to show that a finitely-presented group is word-hyperbolic, has been renamed `IsHyperbolic`. To help overcome the difficulty in using this intrinsic, the Handbook section describing it has been considerably expanded and now includes many more examples. Further, an intrinsic `RedRelatorsForFreeProduct` has been introduced to help preparing input data for certain cases.

10.4 Matrix Groups

10.5 Matrix Groups Over Finite Fields

New Features:

– Fast computation of the conjugacy classes in finite unitary groups is now provided. See the section on Classical Groups for details.

– Major improvements have been achieved in general for the CompositionTree and large matrix groups (LMG) code.

– The computation of orbits has been sped up for matrix groups defined over GF(2).

– The time taken to determine the default choice of base points for the Random-Schreier algorithm has been significantly improved.

– The radical quotient algorithm for BSGS matrix groups, fundamental to many structural computations has been revised to work with all BSGS.

– A number of specialised algorithms for unipotent matrix group structure have been implemented.

10.6 Permutation Groups

Bug Fixes:

– The `quo` constructor now always automatically performs `DegreeReduction` on the resulting permutation group. Previously it did not do so if you also asked for the quotient map (this discrepancy was noted by F. Calegari).

11 Lattices and Quadratic Forms

11.1 Lattices

Changes and Removals:

- The `MinkowskiGramReduction` intrinsic has been internally changed to compute successive minima via `LLL` and `HKZ` rather than call `SuccessiveMinima`. (V2.23-7)

Bug Fixes:

- The third returned value of `LLLGram` was essentially random in the indefinite case, not the rank as specified. (V2.23-3)
- The crash involving `BKZ` intrinsic over the rational field has been fixed. (V2.23-5)
- Some difficulties with spinor genera have been fixed by M. Kirschmer. The problems arise from differences between the notion of Cassels (thus Conway-Sloane) of "spinor genus", which is more often called the proper spinor genus. The `SpinorGenerators` intrinsic now has a `Proper` vararg (by default true) to help with issue. (V2.23-8)
- The `Genus` intrinsic now properly considers denominators.(V2.23-8)
- A crash with insufficient precision in large dimensional real lattices has been changed to give an error message. (V2.23-9)

12 Linear Algebra and Module Theory

12.1 Linear Algebra Over Finite Fields

New Features:

- The base algorithm for matrix multiplication in moderately small dimension has been greatly improved for matrices over GF(q) for $q = 2, 3, 4, 5, 7$. For example, matrix inversion over GF(2) in dimension up to 128 is roughly 3 to 4 times faster on a typical Intel Core CPU.
- The base algorithm for matrix echelonisation has been greatly improved in general for matrices of small to medium size with entries in GF(q) for $q = 2, 3, 4, 5, 7$. For example, matrix inversion over GF(2) in dimension up to 1000 is roughly 3 to 4 times faster on a typical Intel Core CPU.
- Matrix multiplication over GF(p) for $p = 3, 5, 7$ has been significantly improved (for all matrix sizes) in the AVX version.

12.2 Linear Algebra Over General Rings

New Features:

- The recursive echelon algorithm has been improved in the case that a transformation matrix is required. This also yields improvements to many algorithms which depend on this, including the computation of the inverse of a matrix.

- The Hermite normal form algorithm for matrices over the integer ring has been improved for some types of matrix where there are many non-trivial repeated elementary divisors.
- New interpolation-based algorithms have been implemented for computing the determinant and characteristic or minimal polynomial of matrices over a ring R, where R is a integer or polynomial residue ring, or a polynomial ring over such a ring. These are much faster than the previous generic algorithms when applied to matrices of non-trivial sizes.
- The worst-case exponential-time algorithm for computing the characteristic polynomial of a matrix defined over a general commutative ring has been replaced with the Berkowitz algorithm, which takes $O(n^4)$ ring operations for an $n \times n$ matrix. In particular, this speeds up the computation of the characteristic polynomial of a matrix defined over a ring with zero divisors.
- A new generic polynomial-time determinant algorithm has been implemented, based on the Berkowitz algorithm above. This algorithm is applicable to rings with zero divisors for which there is no standard Gaussian elimination-based algorithm available, and is much faster in higher dimensions than the generic exponential algorithm which was previously used.
- New functions `RowWeights` and `ColumnWeights` have been provided for dense matrices.

13 Linear Associative Algebras

13.1 Associative Algebras

New Features:

- A problem with subspaces of empty vector spaces has been fixed. (V2.23-6)

Bug Fixes:

- Constructing sets from orders of algebras has been fixed. (V2.23-6)

14 Representation Theory

14.1 $K[G]$-Modules

New Features:

- The Brooksbank-Luks algorithms for testing modules for isomorphism has been installed. There are three new intrinsics:

  - `SummandIsomorphism`(M, N) determines the unique maximal isomorphic summands of A-modules M and N;
  - `RelativeDecomposition`(M, T) computes the direct sum decomposition of A-module M into $N + K$, where K is minimal with respect to containing the A-module T;
  - `SocleRecursive(M)` finds the socle of the A-module M.

- A new intrinsic `InjectiveHull` which computes the injective hull of a KG-module over a finite field has been installed. This was suggested by Dave Benson and implemented by Derek Holt.
- Intrinsics `Vertex` and `Source` compute a vertex and source (respectively) for a KG-module M, where K a finite field. `Source` also returns the Green correspondent of M. The algorithms used are based on the work of Danz, Kulshammer and Zimmermann.
- Given a KG-module M (usually a projective indecomposable module), the new intrinsic `SocleLayerFactors` determines the composition factors of the socle layers. Here a layer is the quotient of two adjacent terms of the socle series of M. A second intrinsic, `DisplaySocleStructure`, takes the output of `SocleLayerFactors` and displays it in "Christmas tree" style. At present the computation of the socle series employs a naive algorithm but this will be replaced shortly by one that uses condensation.
- The new intrinsic `PIMBlocks` partitions the projective indecomposable KG-modules for the group G and finite field K into blocks. There are two versions of `PIMBlocks`, distinguished by argument types. The first version works off the Cartan matrix which if available is very fast. The second version has as arguments the lists of irreducibles and PIMs and computes the constituents of each PIM. Consequentally, as the dimension of the PIMs increases, this can be very time consuming.

Changes:

- A hang in the intrinsic `IsIsomorphic` for A-modules over finite fields has been fixed. At the same time, the algorithm has been sped up in the case of difficult isomorphism computations for reducible A-modules through use of the Brooksbank-Luks algorithm listed above.

14.2 Basic Algebras

New Features:

- The function `CompactProjectiveResolutionPGroup` has been improved, resulting in a great speedup for some types of basic algebra.

15 System

15.1 Magma Startup

New Features:

– The startup time for Magma has been greatly reduced; it is now typically only 0.02 seconds on a standard Linux installation. This allows many very short successive Magma jobs (called from perl/shell scripts, etc.) to be run much more efficiently.