Summary of New Features in Magma V2.26

April 2021

1 Introduction

This document provides a terse summary of the new features released as part of Magma versions V2.26 (April 2021).

A small number of new features were exported in patch releases prior to the main release of V2.26 in April 2021 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 V2.25-x.

2 Highlights

Arithmetic Geometry

- *Hyperelliptic Curves*

  The functionality for hyperelliptic curves and (smooth) plane quartic curves has been extended by R. Lercier, C. Ritzenthaler, and J. Sijsling. For these classes of curves C, the following new features for hyperelliptic curves have been included:

  - The reconstruction of hyperelliptic curves of genus 2 (respectively 3) from their Igusa invariants (respectively Shioda invariants) has been optimized. The use of separating invariants extends the reconstruction algorithms in genus 3 to all characteristics except 5. Moreover, for all automorphism groups, the resulting reconstructed curves are now returned over the base field whenever possible, and they have small coefficients when the base field is a number field.
  - Given two hyperelliptic curves C_1 and C_2 over a common base field, one can effectively check for the existence of isomorphisms $C_1 \to C_2$ between them, both over the base field and over its algebraic closure. Moreover, the isomorphisms can be computed effectively.
  - Similarly, the automorphism group of a given curve C (over the base field and over the algebraic closure) can be computed. The automorphism group can be obtained as a permutation group.
  - Under some mild conditions on the characteristic, the twists of a given curve C over a finite field, that is, the isomorphism classes of curves that become isomorphic to C over the algebraic closure of the base field, can be calculated effectively.

- *Plane Quartic Curves*

  Some functionality for (smooth) plane quartic curves has been developed by R. Lercier, C. Ritzenthaler, and J. Sijsling similar to that which they have developed for hyperelliptic curves.

  – The Dixmier–Ohno invariants for smooth plane quartic curves (of genus 3) have been implemented, as well as an optimized calculation of their discriminant.

  – An efficient algorithm for reconstructing a smooth plane quartic curve from its Dixmier–Ohno invariants is provided.

  – Given two plane quartic curves C_1 and C_2 over a finite field or the rationals, the existence of isomorphisms $C_1 \to C_2$ can be determined, both over the base field and over its algebraic closure. Moreover, the isomorphisms can be effectively computed.

  – The automorphism group of a given curve C (either over the base field or over the algebraic closure) can be computed. The automorphism group can be obtained as a permutation group.

  – The twists of a plane quartic curve over a finite field can be computed.

- *Elliptic Curves Over Number Fields*

  – Given an order O of a number field K, the intrinsic `EllipticCurveSearch` attempts to find those elliptic curves that have O as conductor. A number of issues that emerged when K is a quartic field have been fixed.

Arithmetic Fields

- *p-adic Rings*

  – A C version of Christopher Doris's package for exact arithmetic in p-adic rings and fields is included in Magma for the first time.

- *Algebraic Number and Function Fields*

  – In 1999, J. Montes introduced a new computational representation, the *OM representation*, for prime ideals in Dedekind domains. This approach for computing with fractional ideals in Dedekind domains has been applied to ideal computations in both number fields and algebraic function fields. A Magma package implementing the Montes approach for both number fields and function fields has been developed by Jens-Dietrich Bauch.

    Part of Bauch's package was incorporated into Magma V2.25 and this has led to dramatic speed-ups in the computation of maximal orders and in the decomposition of primes in number fields and function fields.

This release contains the *OM* representation functionality from Bauch's package. This includes the machinery for ideals and divisors for both number fields and function fields. In particular, this makes possible the construction of ideals and divisors without the computation of maximal orders.

Basic Rings and Fields

- *Univariate Polynomial Rings*
  - Polynomial factorization over finite fields has been greatly sped up when there is a very large number of factors of equal degree.
  - A number of additional speedups have been made which apply generally to the von zur Gathen/Kaltofen/Shoup-based algorithm for polynomial factorization over finite fields.
  - The `&*` operator, when applied to univariate polynomials of the same degree, has been greatly sped up.
  - A parallel version of the Schoenhage-Strassen FFT algorithm for multiplication of polynomials of high degree and with very large integer coefficients has been developed and may be selected with `SetParallelFFT`.
- *Multivariate Polynomial Rings*
  - A major new sparse representation has been developed for monomials of multivariate polynomial rings. As a result, several types of computations involving polynomials with many variables have been sped up and take less memory (particularly when the monomials have very low degree). This includes basic arithmetic, the evaluation of homomorphisms, and the computation of Gröbner bases.

Coding Theory

- *Linear Codes over Finite Fields*
  - POSIX threads and distributed computation are now available for the computation of the minimum weight of codewords. The speed-up factor is approximately equal to the number of cores used.
  - Similar parallelism is available for closely related computations such as the construction of the set of minimum weight codewords.
  - Both POSIX threads and distributed computation type parallelisation can now be used for the computation of the weight distribution and weight enumerator of a linear code. The speed-up factor is close to the number of cores used.

Commutative Algebra

- *Ideal Theory and Gröbner Bases*
  - The function `GroebnerBasis` when applied to an ideal or set/sequence of polynomials now has a second return value D which gives the sequence of degrees encountered in each step of the F_4 algorithm (whether computing the easy GB before a conversion, or a direct F4 computation), when that is non-trivial.
  - The function `EasyIdeal` similarly now has a second return value D giving the sequence of degrees encountered when computing the easy basis (the third return value is now the isomorphism map which used to be the second return value).
  - Major improvements have been made to the algorithm for computing the Hilbert series or Hilbert polynomial of an ideal of a multivariate polynomial ring R, particularly when there are a large number of variables in R.
  - The algorithm for computing the dimension of an ideal of a multivariate ring R and a corresponding maximally independent set has been greatly improved, particularly when there are a large number of variables in R.
  - The sparse case of the F_4 algorithm for inputs defined over $\mathrm{GF}(2^k)$ for $1 < k < 32$ has been improved.
  - The dense F_4 algorithm in 'HFE' mode has been greatly improved when POSIX threads are used. The 80-variable Patarin HFE Challenge 1 can now be solved in 28 seconds on a 4-core 3.6GHz Intel Core i7-7700 processor.
- *Invariant Theory*
  - Some very major improvements in speed and memory usage have been achieved in the linear algebra method for computing the monomials of given degree for an invariant ring (by use of sparse linear algebra where possible).

    This makes possible the computation of properties such as the primary and secondary invariants in the modular case for much larger invariant rings than was previously possible.

    For example, computing the primary invariants of the natural degree-5 representation of $\mathrm{SU}_5(2)$ (with coefficient ring $\mathrm{GF}(2^2)$) is now about 40 times faster than previously.

Group Theory

- *Classical Groups*
  - The algorithm for the conjugacy invariants of classical groups has been improved and as a result there is a significant speed increase in computing the conjugacy classes of classical groups in odd characteristics.
  - Code from Eamonn O'Brien and Giovanni de Franceschi has been installed. Their package computes the conjugacy classes, centralisers and conjugating elements for all classical groups, including groups in characteristic 2.
  - The intrinsic `Classes` (and hence its synonym `ConjugacyClasses`) has been modified so that it uses the new new code when appropriate.
  - The implementation of the orthogonal groups of minus type in even dimensions ($GO^-(2n, q)$ and subgroups) has been changed. For several years, even though the groups produced in different sessions of Magma were isomorphic, the generators were not necessarily the same. In order to address this problem the original C code for the generators has been restored.
- *Finite Groups*
  - The new function `MinimalDegreePermutationRepresentation` developed by Derek Holt takes a finite group G (given as a permutation, matrix, abelian or pc-presentation group) as input, and computes a faithful permutation representation of G of smallest possible degree. The representation and its image, which is isomorphic to G, are returned.

    This calculation can take a long time for some groups G, even when G has faithful permutation representations of moderately small degree. There is an optional parameter `Accept`, which can be set by the user to a positive integer n, in which case the function will stop as soon as it has found some faithful representation of degree at most n and output that. This can speed things up, but of course the result is not guaranteed to have the smallest possible degree in that case.

    The function works quickly on almost all of the groups in the small groups library of groups of order less than 2,000, but it can be slow on a few such examples, particularly large p-groups. These might be suitable candidates for effective use of the `Accept` parameter.

    It generally performs well with matrix groups, provided that a reasonably small degree faithful permutation representation exists. For example, for the group `SpinPlus`(8,3), it finds the optimal representation of degree 4,320 reasonably quickly (about 20 seconds), whereas for `SpinMinus`(8,3), it takes longer (about 180 seconds) to find the representation of degree 183,680 and prove that it has minimal degree.

  Although the new function cannot be used directly on a finitely presented group F, the user can attempt to find an isomorphic permutation group using coset enumeration on F and then apply the minimal degree function.

  – A database containing the 403,874 simple groups of order less than 10^{20} has been installed. The simple groups are arranged in increasing group order thereby making it straightforward for the user to loop though them. The groups can be accessed either by specifying their standard name or their position in the database. While Magma contains tools that allow the user to construct any finite simple group the purpose of this feature is to make it easier for users to access (the smaller) simple groups and to loop through a series of them.

- *Matrix Groups*

  – A new update of the CompositionTree package is included in this release. The handling of representations of $SL(2, q)$ inside `CompositionTree` was revised to ensure that all possible reductions are performed inside `CompositionTree` before calling `RecogniseSL2`; this improves performance for reducible representations of these groups. The intrinsic `CompositionTreeVerify` was revised so that all generators of the input group G are verified to lie in the group H described by the data structure constructed by `CompositionTree`. This additional check is needed to ensure that $H = G$, rather than being just a subgroup of G.

- *Soluble Groups*

  – A package known as *TameGenus* was constructed by Peter Brooksbank, Josh Maglione, and James Wilson as an application of their Multilinear Algebra package. It includes polynomial-time algorithms for deciding isomorphism, constructing automorphism groups, and building canonical labels for a class of finite p-groups of nilpotency class 2 and exponent p. This class includes all d-generator groups of order p^{d+2}, exponent p, and class 2.

Integral Lattices

- *Enumeration in Lattices*
  - Both POSIX threads and distributed computation type parallelisation are now available for the enumeration of shortest vectors. The speed-up factor achieved is approximately equal to the number of cores used.
  - The parallelisation of short vector enumeration automatically parallelises the computation of lattice minimum, packing radius, Hermite constant, centre density, density of the lattice and other computations that involve this type of enumeration.
  - Both POSIX threads and distributed computation type parallelisation are now available for the enumeration of closest vectors. The speed-up factor achieved is close to the number of cores used.
  - The parallelisation of closest vectors enumeration automatically parallelises computations that involve this type of enumeration.

Linear Algebra and Module Theory

- *Linear Algebra over Finite Fields*
  - The algorithms to compute the semilinear dual, the twisted dual and the twisted semilinear dual of a G-module have been updated and as a result there is a considerable speed improvement.
  - The optimal Wilson algorithm for the Gram-Schmidt reduction of the matrix of a reflexive bilinear or sesquilinear form to (almost) diagonal form has been installed.
- *Multilinear Algebra*
  - The calculation of invariant bilinear forms is no longer restricted to finite fields.
  - An upgrade of the package for Multilinear Algebra developed and maintained by Peter Brooksbank, Josh Maglione and James Wilson has been installed. In this upgrade some functions have been optimized and a number of bugs are fixed.

Parallelism

- Several new parallelism facilities are included in Magma V2.26 which are explained in detailed in a new chapter of the Handbook entitled ‘Parallelism’ (Chapter 5). These include the following:
  - The support for user-implemented distributed parallelism has been greatly expanded.
  - POSIX threads and distributed type parallelisation are now available for the enumeration of short vectors and related properties.
  - POSIX threads and distributed type parallelisation are now available for the enumeration of close vectors.
  - The dense F_4 algorithm in ‘HFE’ mode has been greatly improved when POSIX threads are used. The 80-variable Patarin HFE Challenge 1 can now be solved in 28 seconds on a 4-core 3.6GHz Intel Core i7-7700 processor.
  - Multiple NVIDIA GPUs are now supported for matrix multiplication over very small finite fields.
  - POSIX threads and distributed type parallelisation are provided for the computation of the minimum weight and weight distribution of a linear code.

Representation Theory

- *KG-Modules*
  - There is a new function `AbsolutelyIrreducibleModules(G)` for a finite group G to compute all the non-isomorphic absolutely irreducible G-modules in characteristic zero, which are written over $\mathbf{Q}$ or a number field of minimal degree. As an example, the 72 absolutely irreducible G-modules for the group $\mathrm{PSL}(3, 8)$ are computed in about 2 hours, including dimension-511 modules written over a number field of degree 18.
  - The function `IrreducibleModules(G, K)` now supports the cases when K is a quadratic field or a simple number field and returns the non-isomorphic irreducible $K[G]$-modules of a finite group G.
  - The function `GModule(chi)` to compute an individual absolutely-irreducible G-module for an absolutely irreducible character `chi` has been greatly improved for a wide class of inputs.
  - The functions that construct all irreducible and absolutely irreducible KG-modules, where K is a finite field and G is a non-soluble group, have now been extended to include matrix groups over all types of ring. Formerly, it was essentially restricted to permutation groups.

  - The functions that construct all irreducible and absolutely irreducible KG-modules modules, where K is a finite field and G is a soluble group, have been made much more reliable by using a new implementation of the Schur algorithm. (While the former implementation of the Schur algorithm is more general and faster for large groups, it has problems that cause it to fail sometimes).

System

- *NVIDIA GPU support*
  - New functions to control usage of multiple NVIDIA GPUs when present.
  - Multiple NVIDIA GPUs are now supported for matrix multiplication over very small finite fields.

3 Documentation

New Handbook Chapters:

- Parallelism
- Plane Quartics: Invariants and Isomorphisms

4 Language and System Features

New Features:

- New functions `SetNGPUs`, `GetNGPUs`, `SetGPUDevices`, `SetGPUDevice` to control usage of multiple NVIDIA GPUs when present.
- When Magma is started from the shell, the command line option `-t N` may now be used to set the number of POSIX threads to N at startup, where N is a positive integer (so this is equivalent to calling `SetNthreads(N);` after Magma has started).

5 Aggregates and Mappings

Bug Fixes:

- Map application has been fixed so that automatic coercion into the domain is avoided when the input is a sequence of elements in the domain itself. (V2.25-5)

6 Algebraic Geometry

6.1 Schemes

New Features:

- It is now possible to construct a map between a projective scheme and an affine scheme with one of the defining polynomials being zero. (V2.25-5)

Bug Fixes:

- A bug in the intrinsic `Reduction` which computes the reduction of a variety X over a number field K at a place or a prime ideal of K has been fixed. Reported with fix by Raymond van Bommel.
- An error in `PointSearch` which caused a crash when applied to an affine scheme has been fixed.
- A crash in `IsInvertible` applied to a map between schemes when the parameter `Maximal` is set to `true` has been fixed. (V2.25-6)
- An error in the Faugere algorithm which caused the intrinsic `PointsOverSplittingField` (and others) when applied to a scheme to crash has been fixed.

6.2 Algebraic Surfaces

Bug Fixes:

- A bug in the intrinsic `DesingulariseSurfaceByBlowUp` has been fixed.

6.3 Toric Varieties

Changes:

- Maps involving toric varieties have been improved. (V2.25-6)

7 Arithmetic Geometry

7.1 Rational Curves and Conics

Bug Fixes:

– An occasional crash in `HasRationalPoint` for conics defined over a number field has been fixed.

7.2 Elliptic Curves over the Rational Field

Bug Fixes:

– An error in the intrinsic `MordellWeilGroup` caused by a recent change has been fixed.

7.3 Elliptic Curves over Number Fields

Bug Fixes:

– A number of bugs that caused the intrinsic `EllipticCurveSearch` to crash sometimes when the conductor argument is an order in a quartic number field have been fixed.

7.4 Hyperelliptic Curves

In this subsection new intrinsics for hyperelliptic curves are summarised. These were made available in the 2020 version of a package for hyperelliptic curves being developed by R. Lercier, C. Ritzenthaler, and J. Sijsling.

7.4.1 Invariants

New Features:

– `IgusaAlgebraicRelations`: The generators of the ideal of relations between a number of Igusa invariants.

– `IgusaInvariantsEqual`: Determine whether the Igusa Invariants of two genus 2 hyperelliptic curves or of two binary forms of degree 8 are equivalent.

– `DiscriminantFromIgusaInvariants`: The discriminant of a genus 2 hyperelliptic curve given its Igusa invariants.

– `HyperellipticCurveFromIgusaInvariants`: Given the Igusa invariants of a hyperelliptic curve, the curve and its automorphism group are returned.

7.4.2 Automorphisms and Isomorphisms

New Features:

- `IsIsomorphicHyperellipticCurves`: Given two hyperelliptic curves C_1 and C_2, return `true` if they are isomorphic and `false` otherwise.
- `IsomorphismsOfHyperellipticCurves`: Given two hyperelliptic curves C_1 and C_2, return the isomorphisms between C_1 and C_2.
- `AutomorphismsOfHyperellipticCurve`: The automorphisms of the given hyperelliptic curve returned as a list.
- `AutomorphismGroupOfHyperellipticCurve`: The automorphisms of the given hyperelliptic curve returned as a group.
- `GeometricAutomorphismGroupFromShiodaInvariants`: Given Shioda invariants, that is, invariants for a genus 3 hyperelliptic curve C, the intrinsic returns the automorphism group of C.

7.4.3 Twists

New Features:

- `Twists`: Given an elliptic curve, hyperelliptic curve or plane quartic curve, this intrinsic returns the twists of the curve.
- `TwistsOfHyperellipticPolynomials`: Let C be the hyperelliptic curve defined by $y^2 = f(x)$. Given f as argument, this intrinsic returns the twisted hyperelliptic polynomials and their geometric reduced automorphism groups.
- `TwistsOfHyperellipticPolynomials`: Let C be the hyperelliptic curve defined by $y^2 + h(x)y = f(x)$. Given $[f, g]$ as argument, this intrinsic returns the twisted hyperelliptic polynomials and their geometric reduced automorphism groups.
- `TwistedHyperellipticPolynomialsFromShiodaInvariants`: Let C be the hyperelliptic curve defined by $y^2 = f(x)$ and let SI be the Shioda invariants for C. Given SI as argument this intrinsic returns the twisted hyperelliptic polynomials and their automorphism groups.
- `TwistedHyperellipticPolynomialsFromShiodaInvariants`: Let C be the hyperelliptic curve defined by $y^2 + h(x) * y = f(x)$ and let SI be the Shioda invariants for C. Given SI as argument this intrinsic returns the twisted hyperelliptic polynomials and their automorphism groups.
- `TwistsFromShiodaInvariants`: Given the Shioda invariants of a hyperelliptic curve C return the associated twisted hyperelliptic curves and their automorphism groups.
- `TwistsFromIgusaInvariants`: Given the Igusa invariants of a hyperelliptic curve C return the associated twisted hyperelliptic curves and their automorphism groups.
- `TwistsFromG2Invariants`: Given the Cardona-Quer-Nart-Pujolas invariants of a hyperelliptic curve C of genus 2 return the associated twisted hyperelliptic curves and their automorphism groups.

7.4.4 Reduced Isomorphisms and Automorphisms

New Features:

- `IsGL2EquivalentExtended`: Given homogeneous polynomials f_1 and f_2 of degree n where n is at least 4 the intrinsic returns `true` if a matrix T exists such that the change of variable induced on f_1 by T, that is $f_1 * T$, is a multiple of f_2. If there is such a matrix T then all such matrices are returned.
- `IsReducedIsomorphicHyperellipticCurves`: Let C_1 and C_2 be two hyperelliptic curves with defining polynomials f_1 and f_2. Given C_1 and C_2 as arguments, the intrinsic returns a boolean indicating whether a matrix T exists that induces an isomorphism $f_1(x) \to f_2(x)$. f there is such a matrix T then all such matrices are returned.
- `IsReducedIsomorphicHyperellipticCurves`: Let C_1 and C_2 be two hyperelliptic curves with defining polynomials f_1 and f_2. Given f_1 and f_2 as arguments, the intrinsic returns a boolean indicating whether a matrix T exists that induces an isomorphism $f_1(x) \to f_2(x)$. f there is such a matrix T then all such matrices are returned.
- `ReducedIsomorphismsOfHyperellipticCurves`: Let C_1 and C_2 be two hyperelliptic curves with defining polynomials f_1 and f_2. Given C_1 and C_2 as arguments, the intrinsic returns a full list of matrices T that induce an isomorphism $\phi f_1(x) \to f_2(x)$.
- `ReducedIsomorphismsOfHyperellipticCurves`: Let C_1 and C_2 be two hyperelliptic curves with defining polynomials f_1 and f_2. Given f_1 and f_2 as arguments, the intrinsic returns a full list of matrices T that induce an isomorphism $\phi f_1(x) \to f_2(x)$.
- `ReducedAutomorphismsOfHyperellipticCurve`: Given a hyperelliptic curve C, the automorphism group of the defining polynomial of C is returned in the form of a list of matrices.
- `ReducedAutomorphismGroupOfHyperellipticCurve`: Given the hyperelliptic curve C, defined by the equation $y^2 = f(x)$, return the automorphism group of the polynomial $f(x)$ (assumed to be of even degree), as a permutation group.

7.4.5 Point Counting

Bug Fixes:

- An error that occurred sometimes in the intrinsic `IsLocallySolvable` when searching for points on a hyperelliptic curve has been fixed.
- A memory leak in `#` for hyperelliptic curves has been fixed. (V2.25-8)

7.5 Plane Quartic Curves

The functionality for (smooth) plane quartic curves developed by R. Lercier, C. Ritzenthaler, and J. Sijsling is described here in detail.

7.5.1 Basic Operations

New Features:

- `DiscriminantOfTernaryQuartic`: The discriminant of the given ternary quartic form Φ.
- `CovariantHessian`: The Hessian covariant of a ternary quartic form Φ.
- `CovariantSigmaAndPsi`; The covariants Σ and Ψ of a ternary quartic form Φ.
- `QuarticCovariantsAndContravariants`: The generators of the covariant and contravariant algebra associated with a ternary quartic form Φ.

7.5.2 Invariants

New Features:

- `DixmierOhnoInvariants`: The Dixmier–Ohno invariants of the given plane projective quartic curve.
- `DiscriminantFromDixmierOhnoInvariants`: The discriminant of the quartic curve defined by the given Dixmier-Ohno invariants.
- `DixmierOhnoInvariantsEqual`: Given Dixmier-Ohno Invariants $DO1$ and $DO2$ determine whether the invariants are equivalent.
- `DixmierOhnoAlgebraicRelations`: Given the Dixmier-Ohno invariants of a plane quartic curve, this intrinsic returns generators for the ideal generated by relations between the invariants.
- `PlaneQuarticFromDixmierOhnoInvariants`: Given a tuple of Dixmier–Ohno invariants this intrinsic constructs the corresponding plane quartic.
- `TernaryQuarticFromDixmierOhnoInvariants`: Given a tuple of Dixmier–Ohno invariants corresponding to a ternary quartic this intrinsic constructs the quartic.

7.5.3 Isomorphisms and Automorphisms

New Features:

- `IsIsomorphicPlaneQuartics`: Given two plane quartic curves X_1 and X_2, return `true` if they are isomorphic, `false` otherwise.
- `IsomorphismsOfPlaneQuartics`: Given two plane quartic curves X_1 and X_2, return a list of the isomorphisms between them.
- `IsIsomorphicTernaryQuartics`: Given two ternary quartic forms F_1 and F_2, return `true` if they are isomorphic, `false` otherwise.
- `IsomorphismsOfTernaryQuartics`: Given two ternary quartic forms F_1 and F_2, return a list of the isomorphisms between them.
- `AutomorphismsOfPlaneQuartic`: The automorphisms of a plane quartic curve as a sequence of matrices.
- `AutomorphismsOfTernaryQuartic`: The automorphisms of a ternary quartic form as a sequence of matrices.
- `AutomorphismGroupOfPlaneQuartic`: The automorphism group (as an abstract group) of a plane quartic curve.

– `AutomorphismGroupOfTernaryQuartic`: The automorphism group (as an abstract group) of ternary quartic form.

– `GeometricAutomorphismGroup`: Given a elliptic, hyperelliptic, or plane quartic curve this intrinsic computes the geometric automorphism group.

7.5.4 Twists

New Features:

– `TwistsOfPlaneQuartic`: The twists of a given plane quartic curve defined over a finite field.

8 Arithmetic Fields (Global)

8.1 Algebraic Number Fields

New Features:

- The verbose flag `NumberFieldAlgorithms` has been introduced to display when maximal orders, class groups and unit groups are computed as well as information about the orders they are being computed for.

Changes and Removals:

- The implementations of the Round 2 algorithm, Buchmann–Lenstra and Montes algorithms for computing maximal orders have all been improved by using the Smith form of the trace matrix to attempt to compute primes dividing the discriminant more efficiently. (V2.25-6)
- The implementation of the Buchmann-Lenstra algorithm has been further improved by the use of the Dedekind criterion. (V2.25-8)
- `Valuation`s of ideals of orders which are not known to be maximal is now computed by reducing the generators of the ideal by the prime ideal and its powers.
- `Compositum` of two number fields has been made potentially more efficient by checking the smaller degree field for normality first. (V2.25-3)
- Determining whether an element of a number field is an nth power has been made more efficient. (V2.25-5)
- Checking for negative powers of zero has been added to `PowerProduct`.

Bug Fixes:

- Several issues affecting the implementation of the Montes algorithm for computing maximal orders and decompositions of primes have been sorted out. (V2.25-8)
- Using the map returned by `IsIsomorphic` applied to 2 fields when one field is non simple has been fixed. (V2.25-8)
- `Decomposition` of ramified primes in non-maximal orders of Kummer and Artin–Schrier extensions has been fixed. (V2.25-7)
- Error checking has been improved to `meet` for 2 number fields to ensure both are defined by a single polynomial. (V2.25-7)
- A bug in coercion of elements in large towers has been fixed. (V2.25-6)
- A bug in computing `Automorphisms` of a degree 2 field defined by a non-monic or non-integral polynomial has been fixed. (V2.25-7)
- `CoprimeRepresentative` now always returns an element of a field of fractions of an order. (V2.25-4)
- A bug in coercion of number field elements into coefficient rings in a tower has been fixed. (V2.25-5)
- The `LCM` of zero and another element of a quotient of an order has been fixed to return zero. (V2.25-8)
- A crash in the intrinsic `Solutions` when solving certain Thue equations has been fixed. (V2.25-9)

8.1.1 Quadratic Fields

Changes:

- `IsQuadratic` now returns `true` for all degree 2 fields and also returns an isomorphic radical extension as a `FldQuad`. (V2.25-3)

Bug Fixes:

- A bug in `RingClassGroup` for orders of quadratic fields has been fixed. (V2.25-7)
- Conductors of orders of quadratic fields $> 2^{30}$ are now handled better. (V2.25-2)

8.2 Class Field Theory for Number Fields

Changes:

- Checking that extensions are normal and abelian when constructing an `AbelianExtension` has been increased. (V2.25-6)
- The computation of `RayClassGroup`s has been improved by the use of a power product representation of units.

Bug Fixes:

- A crash in `HeckeCharacterGroup` due to equal but non identical orders has been fixed. (V2.25-9)

8.3 Algebraic Function Fields

Changes and Removals:

- `Valuation`s of ideals of orders which are not known to be maximal are now computed by reducing the generators of the ideal by the prime ideal and its powers.
- The implementation of the Buchmann-Lenstra algorithm has been improved by the use of the Dedekind criterion. (V2.25-8)

Bug Fixes:

- A bug when testing `IsIsomorphic` of two curves or function fields has been fixed by F. Hess. (V2.25-7)
- Decomposition of ramified primes in non-maximal orders of Kummer and Artin–Schrier extensions has been fixed. (V2.25-7)
- A bug in coercion of function field elements into coefficient rings in a tower has been fixed. (V2.25-5)
- Several issues affecting the implementation of the Montes algorithm for computing maximal orders and decompositions of primes have been sorted out. (V2.25-8)

8.4 Galois Groups

Bug Fixes:

- A fix has been made to the computation of the `GaloisGroup` of a reducible polynomial over a rational function field over **Q** having zero as a root. (V2.25-7)
- A fix has been made when the `Prime` parameter to `GaloisGroup` is set. (V2.25-7)
- `SolveByRadicals` for polynomials over the rational field has has a recently introduced bug fixed. (V2.25-6)
- The roots returned by `GaloisSplittingField` when given a polynomial over a number field were sometimes wrong. This has been fixed. (V2.25-9)

8.5 Montes Algorithm

The new ideal type `OMIdl` is available for ideals of number fields and function fields. The new divisor type `OMDiv` is available for function fields.

Changes and Removals:

- The implementations of the Montes algorithm for computing maximal orders of number fields has been improved by using the Smith form of the trace matrix to attempt to compute primes dividing the discriminant more efficiently. (V2.25-6)

Bug Fixes:

- Several issues affecting the implementation of the Montes algorithm for computing maximal orders and decompositions of primes have been sorted out. (V2.25-8)

New Features:

- The `Montes` can be called directly for a polynomial or an algebraic field and a prime. The Single Factor Lifting Algorithm can also be called directly on a prime ideal in OM-representation using the intrinsic `SFL`.
- The intrinsic `SetUseMontes` controls whether the Montes algorithm is used by default for maximal order and decomposition calculations in number fields and function fields. Information regarding such setting can be retrieved through `GetUseMontes`.

8.5.1 Ideals in OM-representation

New Features:

- Ideals in OM-representation can be converted to an ideal of type `RngOrdFracIdl` by calling `Ideal`. An OM-representation can be computed from a `RngOrdFracIdl` or from generating elements. using `OMRepresentation`.
- The sum, product, quotient and powers of ideals in OM-representation can be calculated.
- Ideals in OM-representation can be tested for being the one or zero ideal and for equality with other ideals so represented. An element can be tested for being `in` an `OMIdl`. It can be determined whether an `OMIdl` is a `subset` of another such ideal.

– Primality and integrality of ideals in OM-representation can be determined.

– A `pIntegralBasis`, `SIntegralBasis` and `Basis` can be computed for an ideal in OM-representation. Two elements which generate an ideal in OM-representation are turned by `TwoElement`.

– Elements and OM-represented ideals can have their `Valuation` calculated at a prime ideal in OM-representation. Elements can be reduced `mod` such prime ideals and a local expansion computed using `Reduction`.

– The `Factorization` of ideals in OM-representation can be calculated.

– Prime ideals in OM-representation can have their `ResidueField` and `Degree` computed.

8.5.2 Divisors in OM-representation

New Features:

– Divisors in OM-representation can be constructed from one or two `OMIdl` or a function field element using `OMDivisor`.

– The `ZeroDivisor` and `PoleDivisor` of function field elements are divisors in OM-representation.

– A divisor of degree one of a function field can be constructed using `OMDivisorOfDegreeOne`.

– OM-representations can be computed for divisors of type `DivFunElt` using `OMDivisor` and `Divisors` of type `DivFunElt` can be formed from `OMDiv`.

– Divisors in OM-representation can be added, subtracted and multiplied by a scalar. The `GCD` of 2 divisors can be computed.

– Equality of divisors in OM-representation is available. Divisors can be tested for being positive (`IsPositive`) and principal (`IsPrincipal`).

– The `Support` of a divisor in OM-representation is a sequence of prime ideals in OM-representation and a sequence containing the valuations of the divisor at those primes. The `Valuation` of a divisor can be computed at prime ideals in OM-representation.

– Divisors in OM-representation can have their `Height` and `Degree` computed.

– A `Basis` for the Riemann-Roch space of a divisor in OM-representation can be computed along with the `Dimension` of the space. A `ReducedBasis` and a `SemiReducedBasis` are also available.

– The `SuccessiveMinima` and `ApproximatedSuccessiveMinima` of a divisor can be calculated.

9 Arithmetic Fields (Local)

9.1 p-adic Rings and their Extensions

Changes and Removals:

– The computation of `GCD`s during `Factorization` of polynomials over local rings has been made more efficient when one of the polynomials is linear and divides the other.

Bug Fixes:

– A bug affecting `pSelmerGroup`, `PrincipalUnitGroupGenerators` and `ResidueSystem` for local rings containing more than 1 unramified extension has been fixed. (V2.25-6)

– A bug in the construction of a local field which is not a scalar multiple of a monic integral polynomial has been fixed. (V2.25-6)

– Testing `IsNormal` for finite precision rings has been fixed. (V2.25-4)

9.2 Series Rings

Changes and Removals:

– The computation of `GCD`s during `Factorization` of polynomials over local rings has been made more efficient when one of the polynomials is linear and divides the other.

– Improvements have been made to `GCD` computations for polynomials over extensions of series rings.

Bug Fixes:

– Some crashes in `Factorization` with the `Extensions` parameter set to `true` have been fixed. (V2.25-7, V2.26)

10 Basic Rings and Fields

10.1 Integer Ring

Bug Fixes:

- A rare crash in ECPP (coming from `IsPrime`) has been fixed (V2.25-9).

10.2 Finite Fields

New Features:

- The default strategy for selecting the Pohlig-Hellman discrete logarithm algorithm has been improved for general finite fields. Also, if K is a finite field, one may now set the attribute

  ```
  K`UsePHLog
  ```

  (with the backquote operator) to assert whether the Pohlig-Hellman algorithm should or not should be used when computing logarithms in K.

Changes and Removals:

- `Basis` has been fixed to be a relative basis. (V2.25-6)

10.3 Real and Complex Fields

Bug Fixes:

- A bug in computing the complex roots of a polynomial has been fixed. (V2.25-2)

10.4 Univariate Polynomial Rings

New Features:

- Polynomial factorization over finite fields has been greatly sped up when there are a very large number of factors of equal degree.
- The von zur Gathen/Kaltofen/Shoup-based algorithm for polynomial factorization over finite fields has had some further speedups in general.
- The `&*` operator, when applied to univariate polynomials of the same degree, has been greatly sped up.
- A parallel version of the Schoenhage-Strassen FFT algorithm for multiplication of polynomials of high degree and with very large integer coefficients has been developed and may be selected with `SetParallelFFT`.

Bug Fixes:

– A bug in `Normalize` for univariate polynomials defined over integer residue class rings has been fixed, so the result is guaranteed to be unique. (V2.25-6)

– A bug in `EquidimensionalPart` has been fixed (V2.25-5).

– A bug that caused a crash in the intrinsic `Factorization` when applied to a univariate polynomial over a large degree (32) number field has been fixed.

10.5 Multivariate Polynomial Rings

New Features:

– A major new sparse representation has been developed for monomials of multivariate polynomial rings. As a result, several types of computations involving polynomials with many variables have been sped up and take less memory (particularly when the monomials have very low degree). This includes basic arithmetic, the evaluation of homomorphisms, and the computation of Gröbner bases.

– The new procedure `SetSparseMonomialMinRank(R)` sets the limit R so that from now on, whenever a multivariate polynomial ring P is created with at least R variables (i.e., has rank at least R), the sparse representation is used for the monomials of P. The initial value of R is currently 20. Setting R to 0 means that the sparse representation will not be used (so the existing dense representation alone will be used). The corresponding function `GetSparseMonomialMinRank()` returns the current value of R.

– Some automatic coercions are now handled properly with chains of mixed univariate/polynomial rings.

Bug Fixes:

– A crash when computing the variety of a zero-dimensional bivariate ideal over the algebraic closure of a finite field has been fixed.

– The check for overflow in the number of monomials in a multivariate polynomial ring was not quite correct and has been fixed. (V2.25-6)

– Crashes in multivariate polynomial GCD over finite fields have been fixed. (V2.25-7, V2.25-9)

– A hang in polynomial GCD over a certain class of high-degree number fields has been fixed. (V2.25-7)

11 Coding Theory

11.1 Linear Codes over Finite Fields

New Features:

– Both POSIX threads and distributed computation are now available for the computation of the minimum weight and weight distribution of a code. The speed-up factor achieved is approximately equal to the number of cores used. Many associated operations such as the enumeration of all codewords of minimum weight also benefit.

Bug Fixes:

– A crash in `ConstantWords` over large non-prime finite fields has been fixed. (V2.25-8)

12 Commutative Algebra

12.1 Ideal Theory and Gröbner Bases

New Features:

- The function `GroebnerBasis` when applied to an ideal or set/sequence of polynomials now has a second return value D which gives the integer sequence of degrees encountered in each step of the F_4 algorithm (whether computing the easy GB before a conversion, or a direct F4 computation), when that is non-trivial.
- The function `EasyIdeal` similarly now has a second return value D giving the sequence of degrees encountered when computing the easy basis (the third return value is now the isomorphism map which used to be the second return value).
- The algorithm to compute the Hilbert series or Hilbert polynomial of an ideal of a multivariate ring R has been greatly improved, particularly when there are a large number of variables in R.
- The algorithm to compute the dimension of an ideal of an ideal of a multivariate ring R and a corresponding maximally independent set has been greatly improved, particularly when there are a large number of variables in R.
- The F_4 algorithm for inputs defined over GF(2^k) for $1 < k < 32$ in the sparse case has been improved.
- For the dense F_4 algorithm in 'HFE' mode, the whole reduction phase is now completely parallelised when POSIX threads are selected by `SetNthreads`. This is faster than the distinct calls to matrix multiplication in the general dense algorithm and significant memory saving is also achieved. The 80-variable Patarin HFE Challenge 1 can now be solved in 28 seconds on a 4-core 3.6GHz Intel Core i7-7700 processor.

Changes:

- The second return value of `GroebnerBasis(S)` for a set/sequence S is now the third return value and the second return value of `EasyIdeal` is now the third return value; see above for more details.

Bug Fixes:

- A crash when computing large Groebner bases over non-prime finite fields of characteristic 2 has been fixed. (V2.25-6)
- Incorrect sign handling when computing coordinates of polynomial/vectors in ideals/modules defined over exterior algebras has been fixed. (V2.25-6)
- A missing error check for noncommutative Groebner basis computation over unsupported rings has been added. (V2.25-6)
- A bug in `Variety` for ideals defined over algebraic closures has been fixed. (V2.25-8)
- A crash in computing Groebner bases over the field GF(2^k) for $k = 2, 3, 4$ has been fixed. (V2.25-9)
- A crash in computing primary decomposition over algebraically closed fields has been fixed. (V2.25-9)

12.2 Modules over Multivariate Polynomial Rings

New Features:

- The algorithm to compute the Hilbert series or Hilbert polynomial of a module over a multivariate ring R has been greatly improved, particularly when there are a large number of variables in R.

12.3 Invariant Theory

New Features:

- Some very major improvements in speed and memory usage have been achieved in the linear algebra method for computing the monomials of given degree for an invariant ring (by use of sparse linear algebra where possible).

  This now allows the computation of properties such as the primary and secondary invariants in the modular case for much larger invariant rings than were previously possible.

  For example, computing the primary invariants of the natural degree-5 representation of $\mathrm{SU}_5(2)$ (with coefficient ring $\mathrm{GF}(2^2)$) is now about 40 times faster than previously.

13 Groups

13.1 Classical Groups

New Features:

- The algorithm to compute the conjugacy invariants of all classical groups in odd characteristic has been updated and as a result there is a significant speed increase in computing the conjugacy classes of classical groups.

  Thanks to new code from Eamonn O'Brien and Giovanni de Franceschi it is now possible to compute the conjugacy classes, centralisers and conjugating elements for all classical groups, including groups in characteristic 2. The new intrinsics include `ClassicalIsConjugate`, `ClassicalCentraliser`, `ClassicalClassMap`, `ClassesForFixedSemisimple` and `IsometryGroupClassLabel`.

Changes:

- The intrinsic `Classes` (and hence its synonym `ConjugacyClasses`) has been modified so that it uses the new code when the input is a classical group.
- The implementations of the intrinsics `GOMinus`, `SOMinus` and `OmegaMinus` have been changed. The previous implementations can be accessed via the intrinsics `AltGOMinus`, `AltSOMinus` and `AltOmegaMinus`. The groups returned by `GOMinus(n,q)` and `AltGOMinus(n,q)`, for example, will be isomorphic but the generators returned for `AltGOMinus(n,q)` may vary from session to session.

Bug Fixes:

- Two bugs in the code for GLNormaliser which caused it to crash have been fixed.

13.2 Finite Groups

Additions:

- The new intrinsic `MinimalDegreePermutationRepresentation` developed by Derek Holt takes a finite group G (of type `GrpPerm`, `GrpMat`, `GrpAb` or `GrpPC`) as input, and computes a faithful permutation representation of G of smallest possible degree. The representation and its image, which is isomorphic to G, are returned.

13.3 Matrix Groups Over Finite Fields

Changes:

- The handling of representations of $SL(2,q)$ inside `CompositionTree` has been revised to ensure that all possible reductions are performed before calling `RecogniseSL2`.
- The intrinsic `CompositionTreeVerify` was revised to ensure that all generators of the input group G lie in the group H defined by the `CompositionTree` data structure.
- The code which converted the return value of tt SpinorNorm to the Zassenhaus convention has been removed. This ensures that in characteristic not 2 the spinor norm of an element g of the orthogonal group is consistent with `SpinorNorm(V,g)`, where V is a quadratic or inner product space.

13.4 p-Groups

New Features:
The *TameGenus* package developed by P.A. Brooksbank, J. Maglione, and J.B. Wilson provides fast algorithms for computing automorphism groups and deciding isomorphism for "tame genus" p-groups.

- If G is a finite p-group of tame genus, the intrinsic `TGAutomorphismGroup` returns the automorphism group of G.
- If G and H are finite p-groups of tame genus, the intrinsic `TGIsIsomorphic` returns *true* if G and H are isomorphic.
- If G is a finite p-group, the intrinsic `IsTameGenusGroup` decides whether `TGAutomorphismGroup` and `TGIsIsomorphic` can be applied to G.

13.5 Databases of Groups

New Features:

A database which currently comprises the 403,874 simple groups of order less than 10^{20} has been installed. While Magmacontains tools that allow the user to construct any finite simple group, the purpose of this facility is to make it easy to iterate through a selection of simple groups. When reading the descriptions of the intrinsics below it is important to note that a simple group may be specified in one of three ways:-

- A string giving the standard name of the simple group.
- An internal description of the simple group consisting of a tuple containing three integers.
- The number of the simple group in the database.

- The intrinsic `SimpleGroup` returns a simple group isomorphic to the specified simple group. The group may be specified by any one of the above three methods.
- The intrinsic `SimpleGroupOfOrder` returns a simple group of the indicated order. The order may be given either as an integer or as a factorisation sequence containing the factored order.
- The intrinsic `NumberOfSimpleGroups` returns the number of simple groups present in the database.
- Given the database number of a simple group the intrinsic `SimpleGroupName` returns the standard name of the simple group.
- Given the standard name of a simple group the intrinsic `SimpleGroupNameToNumber` returns the number of the simple group in the database.
- Given the internal descriptor of a simple group the intrinsic `SimpleGroupIdToNumber` returns the number in the database of the simple group specified by the internal descriptor.
- Given the database number of a simple group the intrinsic `SimpleGroupId` returns the internal descriptor as a tuple of three integers.

14 Lattices and Quadratic Forms

14.1 Lattices

New Features:

- Both POSIX threads and distributed computation type parallelisation are now available for the enumeration of shortest vectors. The speed-up factor achieved is approximately equal to the number of cores used.
- The parallelisation of short vector enumeration automatically parallelises the computation of lattice minimum, packing radius, Hermite constant, centre density, density of the lattice and other computations that involve this type of enumeration.
- Both POSIX threads and distributed computation type parallelisation are now available for the enumeration of closest vectors. The speed-up factor achieved is close to the number of cores used.
- The parallelisation of closest vectors enumeration automatically parallelises computations that involve this type of enumeration.
- The integral variant of the LLL algorithm (selected by the parameter `Method := "Integral"`) now has POSIX-thread support (selected by `SetNthreads`).
- The algorithm for `Coordinates` for lattice elements defined over a real field has been made more robust.
- New functions `ShortVector, ShortestVector, CloseVector, ClosestVector` to return a single non-zero short/close/shortest/closest vector in a lattice.

Bug Fixes:

- A bug giving incorrect results in the maps for `DualQuotient` has been fixed. (V2.25-3)
- Setting one of the attributes `Minimum`, `Kissing Number`, or `Theta Series` when one of the other attributes is set will no longer print the theta series. (V2.25-6)
- Bugs in `IsCoercible` and `Coordinates` for lattice elements defined over a real field have been fixed.

14.2 Lie Algebras

New Features:

- The closure algorithm for the computation of subalgebras of Lie algebras has been greatly improved, particularly in the case of large sparse algebras in characteristic zero.

15 Linear Algebra and Module Theory

15.1 Matrices

New Features:

- Multiple NVIDIA GPUs are now supported for matrix multiplication over very small finite fields.

Changes:

– `AddRow` and `AddColumn` now accept a `RngElt` as a scalar, and not only an integer.

Bug Fixes:

– An issue where the multiplication of two large integer matrices could incorrectly compute the resulting size (leading to memory exhaustion) has been fixed. (V2.25-4)

– A bug when computing the determinant of matrices over polynomial rings over non-domains has been fixed. (V2.25-6)

– A crash in the integer Hermite algorithm (arising from computing the map arising from calls by `AbelianQuotient`) has been fixed. (V2.25-7)

– A bug that occurred when taking projections of subspaces of a vector space over the rationals has been fixed. (V2.25-7)

New Features:

– The efficiency of the function `Submatrix(S, I, J)` where S is a sparse matrix and I, J are index sequences has been greatly improved (by writing an inefficient internal version).

15.2 Bilinear and Sesquilinear Forms

New Features:

– The intrinsic `GramSchmidtPair` performs a Gram-Schmidt reduction of the matrix of a reflexive bilinear form using a new optimal algorithm due to James Wilson.

– The intrinsic `IsReflexive` determines whether a matrix represents a reflexive form; if it does, the type of the form and the multiplier are returned.

– `EnsureUpperTriangular` returns returns the upper triangular matrix defining the same quadratic form as a given matrix.

Changes:

– The number of values accepted by the `Variant` parameter of the intrinsics `StandardQuadraticForm` and `StandardSymmetricForm` has been reduced from three to two: `"Default"` and `"Revised"`. This is for consistency with the changes to the generators of the even dimensional orthogonal groups of minus type. The `"Revised"` variant returns the form preserved by the groups constructed by `AltGOMinus`, `AltSOMinus` and `AltOmegaMinus`.

15.3 Vector Spaces

New Features:

– The `*:=` operator has been improved for vector space elements.

15.4 Multilinear Algebra

New Features:

- The intrinsic `MatrixTensor` returns the K-tensor given by rectangular matrix multiplication.
- A subspace of bilinear tensors can be multiplied on the left or right to give a new subspace.
- The composition of `InducedTensor` and `AsMatrices` is implemented efficiently in `SliceAsMatrices`.
- The intrinsic `DerivedFrom` will include some tensor data in a given matrix object,
- Now that `HeisenbergGroup` returns a `GrpMat`, `HeisenbergGroupPC` is provided to return a `GrpPC`.
- Additional random tensors can be generated by `RandomCotensor`, `RandomAlternatingTensor`, `RandomAntisymmetricTensor` and `RandomSymmetricTensor`.
- The intrinsic `IsHomotopism` decides whether given maps form a homotopism.
- A tensor can be `Precompose`d with a map in a given coordinate.
- The `SelfAdjointAlgebra`s for a tensor can be constructed.
- More signatures are provided for `Homotopism`, `Centroid` and `DerivationAlgebra`.
- The `.` operator can be used on a homotopism `Hmtp` to gain the map on a coordinate. A homotopism can now be applied `@` to a tensor.
- The `Valence` of a homotopism can be retrieved. Several `Shuffle`s of a homotopism can be computed.

16 Linear Associative Algebras

16.1 Associative Algebras

Bug Fixes:

- Incorrect printing of elements in Magma mode has been fixed.

16.2 Quaternion Algebras

Bug Fixes:

- A bug when computing of the discriminant of a quaternion algebra has been fixed by J. Voight. (V2.25-6)

17 Representation Theory

17.1 $K[G]$-Modules

New Features:

- There is a new function `AbsolutelyIrreducibleModules(G)` for a finite group G to compute all the non-isomorphism absolutely irreducible G-modules in characteristic zero, which are written over **Q** or a number field of minimal degree. As an example, the 72 absolutely irreducible G-modules for the group $\mathrm{PSL}(3, 8)$ are computed in about 2 hours, including dimension-511 modules written over a number field of degree 18.
- The function `IrreducibleModules(G, K)` now supports the case that K is a quadratic field or a simple number field and returns the non-isomorphic irreducible $K[G]$-modules of a finite group G.
- The function `GModule(chi)` to compute an individual absolutely-irreducible G-module for an absolutely irreducible character `chi` has been greatly improved for a wide class of inputs.
- The functions `IrreducibleModules(G, K)` and `AbsolutelyIrreducibleModules(G, K)`, where G is a non-soluble group and K is a finite field, have now been extended to include matrix groups over all types of ring. Formerly, it was essentially restricted to permutation groups.

Changes:

There is an important change when computing irreducible KG-modules, where K is a finite field and G is a matrix group:

- The functions `IrreducibleModules(G, K)` and `AbsolutelyIrreducibleModules(G, K)`, where G is a soluble group and K is a finite field, have been made much more reliable by using a new implementation of the Schur algorithm.
- For irreducible modules use `IrreducibleModules(G, K)` which will return the irreducible modules for G over the finite field K.
- For absolutely irreducible modules use `IrreducibleModules(G, K)` or `AbsolutelyIrreducibleModules(G, p)`, where p is the characteristic of K. This returns the absolutely irreducible modules for G in characteristic p (written over $\mathrm{GF}(p^k)$ for minimal k in each case).

Bug Fixes:

- A bug has been fixed which caused functions to compute irreducible modules for a group over a finite field (such as `IrreducibleModules` and `AbsolutelyIrreducibleModules`) to crash in certain circumstances. (V2.25-7)
- A bug in the code for the function `IrreducibleModulesSchur` which occurs when computing KG-modules, K a finite field and G a soluble group, has been fixed. (V2.25-7)
- A bug in the characteristic-0 Meataxe which caused it to hang in the case of some small KG-modules has been fixed. (V2.25-7)
- A bug in the characteristic-0 Meataxe which caused it to crash when splitting a module over a cyclotomic field has been fixed. (V2.25-7)

- An incorrect result for the function `IndecomposableSummands` (or `DirectSumDecomposition`) applied to a $\mathbf{Q}G$-module where G is a soluble group has been fixed. (V2.25-8)
- A crash in `IsIsomorphic` for G-modules over finite fields has been fixed. (V2.25-9)
- The `IsIsomorphic` algorithm for G-modules over finite fields has been improved, so that the initial tests for non-isomorphism are much stronger, fixing slow behaviour for certain inputs. (V2.25-9)
- A crash in intrinsic `IrreducibleModules` when applied to a KG-module, where K is the rationals, has been fixed. (V2.25-9)
- A crash in intrinsic `Constituents` when applied to a KG-module, where K is a cyclotomic field, has been fixed. (V2.25-9)