Summary of New Features in Magma V2.19

December 2012

1 Introduction

This document provides a terse summary of the new features installed in Magma for release in version V2.19 (December 2012). A small number of new features were exported in patch releases prior to the main release of V2.19 in December 2012 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 V2.18-x.

Recent releases of Magma were: V2.18 (December 2011), V2.17 (December 2010), V2.16 (November 2009), V2.15 (December 2008), V2.14 (October 2007), V2.13 (July 2006).

2 Highlights

Algebraic Geometry

- *Schemes*
  - A package of functions is provided for working with divisors on varieties. In the case of surfaces some additional functionality is available. The package includes decomposition into irreducible components, Riemann-Roch spaces, canonical divisors and (surface) intersection numbers. The package was developed by Martin Bright, Gavin Brown, Mike Harrison and Andrew Wilson.
- *Curves*
  - Code implementing gonal maps and small-degree plane models for curves of genus less than 7 has been installed. Functions are provided which construct the smallest degree (gonal) maps of the curve onto the projective line (equivalent to the smallest degree functions). This machinery applies to all curves of genus less than 7 as well as hyperelliptic and trigonal curves of any genus. Construction of birationally-equivalent plane curves of smallest degree for most classes of non-hyperelliptic curves of genus less than 7 is supported.
  - Functions to compute Shioda invariants for genus 3 hyperelliptic curves, reconstruct models for a curve from such invariants and compute geometric automorphism groups are included. The package was written and made available by Reynald Lercier and Christophe Ritzenthaler.
  - A function which gives the complete list of intersection multiplicities for all intersection points of two plane curves over the rationals in a single calculation

following the algorithm of Hilmar and Smyth will appear in an early patch release of V2.19, adapted into Magma from code provided by Chris Smyth.

- *Surfaces*

  Magma V2.19 introduces an important collection of tools for identifying the type of a surface and computing its basic invariants.

  - The Kodaira dimension and precise Kodaira-Enriques type (e.g. K3, bi-elliptic) as well as basic invariants (irregularity, "Hodge diamond" numbers) can be calculated for projective surfaces having only simple singularities.
  - Minimal models can be computed for projective surfaces having only simple singularities. For rational surfaces, this gives the standard not-quite-minimal models (Del Pezzos, scrolls, ...). For Kodaira dimension 1, a map to a non-singular curve which presents the minimal surface as an elliptic fibration can also be computed.
  - In the case of projective surfaces of general type having only simple singularities, the full canonical coordinate ring can be calculated.
  - A test is provided to determine whether an isolated singular point of a scheme is analytically equivalent to a hypersurface singularity. The equivalent hypersurface equation may be computed to desired precision. It is possible to test whether a surface has only simple (A-D-E) singularities.
  - As well as the test for only simple singularities, there is a test for normality. Tests for more general schemes being Gorenstein, Cohen-Macaulay or the arithmetic versions of these when in ordinary projective space.
  - Intrinsics are provided to generate random (non-singular) surfaces from many of the families of surfaces in 4-dimensional projective space described by Decker, Ein, Schreyer and Ranestad.
  - Code that calculates the invariants, covariants and contravariants of a cubic surface has been developed by Andreas-Stephan Elsenhans.

- *Toric Geometry*

  - Algorithms have been implemented for computing the Demazure roots of a complete fan. This capability is then used to implement tests as to whether a complete fan is semistable or reductive. The Demazure roots of the associated fan can also be used to determine whether the complete toric variety is isomorphic to a (product of) projective space.

Arithmetic Fields

- *Number Fields*
  - A new implementation of the standard subexponential algorithm for class groups of general number fields has been written by Steve Donnelly. This replaces the old version of the standard algorithm. The main advantages are as follows.
    * Reliability – running times are now quite predictable; previously, in a significant subset of "awkward" cases, the computation would get stuck for long, highly random periods.
    * Large degree – the old implementation generally failed to terminate for fields of degree above 20 or so, even when the discriminant was not so large that the conditional class group is inherently difficult. Degrees in the range 20 to 40 or more are now routinely handled.
    * Speed (in forthcoming releases) – it is anticipated that running times will be several times faster, for most values of *(degree, discriminant)* once bottlenecks elsewhere are removed (this will be done during the coming year).

    The gains are due to some refinements to the standard algorithm, better techniques for some low-level steps, and (perhaps primarily) a very careful strategy which relies in part on estimating the expected cost.
  - An estimate can be obtained for the expected cost of computing the class group (under GRH) of a given field using the standard algorithm. The estimate matches reality up to a modest factor (say 20–30%) for most values of *(degree, discriminant)*.
  - The sieve algorithm for class groups of general number fields has been sped up.
- *Global Fields*
  - The computation of coercions between orders and fields has been made more efficient saving both significant memory space and time in some examples.
- *Characters and Artin Representations*
  - Intrinsics have been added for computing the central character for Dirichlet and Hecke characters (including Grössencharacters). Similarly, the determinant of an Artin representation is returned as an Artin representation.

Arithmetic Geometry

- *Elliptic Curves Over* $\mathbf{Q}$
  - The current Magma version of the database of elliptic curves over $\mathbf{Q}$ constructed by John Cremona contains all curves having conductor up to 300,000.
  - Given the real and imaginary periods of a minimal model for an elliptic curve over $\mathbf{Q}$, it is now possible to construct the curve.
- *Elliptic Curves Over Function Fields*
  - A routine to find elliptic curves with given conductor has been added.
- *Hyperelliptic Curves*
  - Euler factors at good primes are now computable for hyperelliptic curves over number fields.
  - There have been several improvements to the machinery for computing rank bounds for Jacobians of hyperelliptic curves, including improved implementations of 2-descent and an algorithm of Creutz to obtain better upper bounds for the Mordell-Weil rank.
  - New intrinsics have been added to allow one to determine the index of a hyperelliptic curve over a local field.
- *Cyclic Covers of* $\mathbf{P}^1$
  - Descents on Jacobians and Pic^1 torsors of cyclic covers of $\mathbf{P}^1$ using algorithms of Poonen-Schaefer and of Creutz are now available.
  - Descents on cyclic covers of $\mathbf{P}^1$ with singular models using an algorithm of Mourao have been made available.
- *Hypergeometric Motives*
  - A new package for hypergeometric motives has been implemented. The main functionality is the computation of the hypergeometric traces (via p-adic Γ-functions), which give Euler factors of a (presumed) L-series. This is then linked to the L-series package.

    Other aspects of the package include the ability to recognise various special hypergeometric motives (or data) in terms of Artin representations or (hyper)elliptic curves.

Arithmetic Geometry (Modular Forms)

- *Hilbert Modular Forms*
  - It is now possible to directly compute the newforms that have rational Hecke eigenvalues (without computation of newspaces or decomposition).
- *Brandt Modules*
  - A new implementation of Brandt modules associated to definite quaternion orders, over $\mathbf{Z}$ and also over function fields $\mathbf{F}_q[t]$, has been developed by Markus Kirschmer and Steve Donnelly. The polynomial case relies on algorithms for quadratic forms developed by Kirschmer. The approach is similar to that used in the package for Hilbert modular forms (the definite case); this has a huge advantage over the old Brandt module implementation over $\mathbf{Z}$, in the case where the Eichler level is not very small. The main features implemented are: dimensions via formulae, Hecke operators, decomposition, and efficient computation of newforms and systems of eigenvalues.

Associative Algebras

- *Basic Algebras*

  - New functionality for computing with homomorphisms of basic algebras has been added. A homomorphism for a basic algebra can be constructed from a matrix of appropriate size and such a map or matrix can be tested to see if it is a homomorphism of algebras. The image of a homomorphism is returned as a subalgebra together with the inclusion homomorphism.
  - The automorphism group of a basic algebra can be computed and two basic algebras can be tested to see if they are isomorphic. The implementation generalizes an algorithm of Bettina Eick for nilpotent structure constant algebras.
  - Functionality to construct ideals, subalgebras and quotient algebras has been added. The subalgebra is returned as a basic algebra together with the inclusion homomorphism. The returned subalgebra includes the minimal idempotent that acts as an identity on the given elements.
  - Functions have been added to construct standard subalgbras such as the centre of a basic algebra and the centralizer of a collection of elements in a basic algebra.
  - There are new functions to construct the associated graded algebra of a basic algebra and a standard form (not unique) for a basic algebra. Included are new functions to create the basic algebra of the action algebra on the direct sum of a sequence of modules over a matrix algebra or group algebra. An example is the basic algebra of a block of a group algebra.

- *Matrix Algebras*

  - For a matrix algebra defined over a finite field and having a unit element, faster algorithms for computing a basis, a basis of the Jacobson radical and the unit group have been added.

- *Quaternion Algebras*

  - An improved algorithm for computing the two-sided ideal classes of an order in a definite quaternion algebra (over $\mathbf{Z}$ or $\mathbf{F}_q[t]$) has been developed and coded by Markus Kirschmer. The algorithm relies on being able to efficiently compute the normalizer of an order.

Basic Rings and Fields

- *Finite Fields*

  – This release includes a major new implementation by Allan Steel of the Coppersmith index-calculus algorithm for computing discrete logarithms in finite fields of small characteristic. For characteristic greater than 2, a straightforward generalization of Coppersmith's algorithm is used. This means that discrete logarithms can now be quickly computed for many fields of small characteristic for the first time.

    The performance is greatly enhanced by pre-computing auxiliary tables so that the initial computation of the logarithms of a factor base can be avoided. This means that logarithms of individual elements can be computed immediately if the relevant table is present for the field under consideration. Tables are currently available for most finite fields of characteristic 2, 3, 5 or 7 with cardinality up to 2^{400} and also for many non-trivial fields of higher characteristic.

- *Finite Near-Fields*

  – This release includes a new package implementing finite near-fields and related finite projective (non-Desarguesian) projective planes. Both ordinary and exceptional near-fields can be constructed. Functions are provided for computing the group of units, the automorphism group and for testing isomorphism of two near-fields.

Coding Theory

- *Linear Codes over Finite Fields*

  – Using their Toric Geometry machinery, Gavid Brown and Al Kasprzyk have implemented intrinsics for constructing the linear code over a finite field associated with the lattice points in a polytope P.

Combinatorial Theory

- *Matrices*
  - It is now possible to compute the group of row and column permutations of a matrix M that fixes M. The corresponding isomorphism test is also available. A very efficient `nauty` algorithm of B. McKay is used.

Commutative Algebra

- *Gröbner Bases*
  - The performance of the algorithm for computing a Gröbner basis over **Z** has been greatly sped up by using modular techniques where possible.
  - By using more intelligent pair selection, the computation of a Gröbner basis over an Euclidean ring has been made much more efficient.
  - The algorithm that finds the primary decomposition of a zero-dimensional ideal over **Q** been made much faster for several classes of inputs.

Geometry

- *Convex Polytopes*
  - A version of the PALP normal form has been implemented in Magma by Al Kasprzyk. This is the first implementation of the famous algorithm of Max Kreuzer and Harald Skarke outside of PALP. An alternative algorithm has also been developed which is significantly faster when the polytope has symmetries.
  - The affine normal form of the maximum dimensional lattice polytope can now be calculated.
  - The Minkowski decompositions of polytopes in arbitrary dimensions can now be computed. The algorithm is based on a result of Klaus Altmann.
- *Finite Geometry*
  - As a consequence of the recent implementation of finite near-fields, constructions for the non-Desarguesian projective plane coordinatised by a near-field and the Hughes plane have been implemented.

Group Theory

- *Finitely-Presented Groups*
  - An improved version of the Plesken-Fabianska algorithm for computing L2-quotients of a 2-generator finitely presented group developed and implemented by Sebastian Jambor has been installed. Given a finitely presented group G on two generators, the algorithm simultaneously computes all quotients of G which are isomorphic to some $\mathrm{PSL}(2, q)$, for any prime power q. It can handle the case of infinitely many quotients, and also works for very large prime powers.
  - The machinery for computing with Automatic Groups has been extended to provide access to information regarding the underlying automata and to allow automatic groups to be written to, and read from, files. A database of automatic groups corresponding to 4000 hyperbolic 3-manifolds from Weeks' census of hyperbolic 3-manifolds of small volume will shortly be made available.
- *Matrix Groups (Over Finite Fields)*
  - The package for constructing the Composition Tree representation of a matrix group has been extended as follows:

    (i) Short presentations on standard generators for the orthogonal groups have been installed. As a consequence, short presentations on standard generators are now available for all families of classical groups.

    (ii) A new function has been provided for writing an element of a classical group G as an SLP in the standard generators of G. The algorithm to perform this task for a classical group in defining characteristic was developed by Elliot Costi and the black box equivalent is due to Csaba Schneider. Schneider prepared the code for this function which uses Costi's implementation in the case of the natural copy.

    The availability of presentations enables rapid verification of the Composition Tree for any group whose composition factors exclude exceptional groups of Lie type. It is planned to add short presentations for the exceptional groups in the near future.
  - Lifting-style algorithms have been developed by Derek Holt for new tasks in groups given in terms of the Composition Tree data structure. These include determining centralisers, conjugacy of elements, computing conjugacy classes, normalizers, subgroup conjugacy and maximal subgroups.
- *Matrix Groups (Over Z and Q)*
  - A much improved algorithm for computing the normaliser or centraliser of a finite subgroup of $GL(n, Z)$ has been implemented by Markus Kirschmer. A slight variation of the algorithm can be used for testing conjugacy of finite subgroups of $GL(n, Z)$.

- Opgenorth's algorithm has been adapted by Markus Kirschmer to provide a test for the conjugacy of two matrices of finite order in $GL(n, Z)$. A slight variation provides an effective method for computing the centraliser of an element of finite order.
- An algorithm for determining the conjugacy of any pair of matrices in $GL(2, Z)$ was developed by D. Husert (University of Paderborn). In particular, this allows the conjugacy of elements having infinite order to be determined. Code developed by Husert is now included in Magma.

- *Permutation Groups: Databases*

  - It is now possible to identify a transitive permutation group of degree 32 in the database of all transitive groups of degree 32; this requires the installation of an optional database containing identification data.

- *Quasisimple Groups*

  - The construction of all irreducible representations of quasisimple groups up to some fixed degree has been a goal for some time. In 2001, Hiss and Malle published a list of the possibilities up to degree 248. Derek Holt has constructed a table of modular representations of all such groups up to degree 100. This table can now be accessed from within Magma.

- *Soluble Groups*

  - A package for the more efficient calculation of the automorphism group of a finite soluble group G is included. The algorithm was developed by David Howden and works by extending the automorphisms of a large Sylow p-subgroup of G to the automorphism group of G. A slight variation of the algorithm may be used to test isomorphism.

Lattices and Quadratic Forms

- *Integral Lattices*
  - It is now possible to test a pair of quadratic forms or a pair of lattices for being rationally equivalent.
  - Code has been developed to compute the subgroup of the automorphism group of a lattice which also fixes a collection of forms setwise. The forms are not required to be positive definite or even symmetric. This is a variation of the partition-refinement backtrack lattice automorphism program developed by Bill Unger in 2010.

Lie Theory

- *Lie Algebras*
  - A major revision of the basic Lie Algebra machinery has resulted in major performance improvements, especially in the case of larger dimensional algebras. Firstly, the basic multiplication algorithm for elements of a structure constant Lie algebra has been sped up. Secondly, the creation of subalgebras and quotient algebras is now much faster in general. Thirdly, the algorithm used to compute a composition series is now much more efficient. In particular, this makes it possible, for the first time, to compute composition series for algebras having dimensions well into the thousands. It is also possible to determine the composition factors without explicitly computing a composition series in much less time than it takes to find a composition series.
  - An intrinsic to compute the Plesken Lie algebra of a finite group G is now available. This is the linear span of the elements $g - g^{-1}$ in the group algebra of G.
  - Support for directed W-graphs has been implemented (code supplied by Bob Howlett).

Linear Algebra and Module Theory

- *Linear Algebra*

  - A revised implementation of the Lanczos algorithm for sparse matrices over finite fields provides greatly improved performance, particularly for larger matrices.
  - The Hermite Normal Form algorithm is now much more efficient for the case of sparse matrices over **Z** and **Q** having full rank and which have at least one large elementary divisor. This benefits algorithms such as the relation method for computing ideal class groups of number fields.

- *Bilinear and Sesquilinear Forms*

  - Facilities for computing with vector spaces equipped with a bilinear, sesquilinear or quadratic form have been added. Isometry and similarity groups can be computed and the spaces of forms which are invariant or semi-invariant under the action of a matrix group can be found.

Representation Theory

- *Galois Representations*

  - A package has been developed by Jeremy Le Borgne which provides tools for working with φ-modules over the power series ring $k((u))$, where k is a finite field, and representations of the absolute Galois group of $k((u))$ with coefficients in a finite field.

    The main functionality is concerned with computing the semisimplification of a given φ-module, and the semisimplification of the Galois representation that is naturally attached to it. In particular, the slopes of the φ-module, corresponding to the tame inertia weights of the Galois representation, can be computed using this package.

System

- *Language*
  - User-defined types are now available for the first time. This facility allows the user to declare new type names and create objects with such types and then supply some basic primitives and intrinsic functions for such objects.

3 Documentation

New Handbook Chapters:

- Nearfields
- Polar Spaces
- Hypergeometric Motives
- Mod p Galois Representations

4 Language and System Features

New Features:

- User-defined types are now available for the first time. This facility allows the user to declare new type names and create objects with such types and then supply some basic primitives and intrinsic functions for such objects.

Changes and Removals:

- When `SetQuitOnError(true)` is in force, the exit status is now non-zero when Magma exits due to an error.

Bug Fixes:

- Fixed a possible crash when a syntax error arises within a function definition.
- Fixed a crash that could arise when using `eval` with a database around.
- The algorithm deciding the frequency of garbage collection has been revised so as to spend much less time in some situations.

5 Aggregates and Mappings

New Features:

- The operator `#` now works for associative arrays (returning the number of keys).

Changes and Removals:

- The speed of `RandomSubset` has been improved for large sets.

Bug Fixes:

- The functions `And`, `Or`, `Xor`, and `Not` no longer crash when one of the arguments is the null sequence.
- A bug has been fixed which could cause hashing of multisets to return invalid values for certain underlying types such as vectors.
- A bug in associative arrays that could cause crashes when keys of different universes are used has been fixed.
- A memory leak which occurred when records are created has been fixed.

6 Algebraic Geometry

6.1 Schemes

New Features:

- Tests have been added for whether an ordinary projective scheme is Gorenstein, Cohen-Macaulay or the arithmetic versions of these: `IsGorenstein`, `IsArithmeticallyGorenstein` and the same with `Gorenstein` replaced by `CohenMacaulay`.
- A package of code to work with divisors on varieties has been added. This is in an early stage of development but contains a range of intrinsics to create divisors, decompose into primary or prime components, work with **Q**-rational divisors, perform basic arithmetic, compute Riemann-Roch spaces, test for linear equivalence, compute intersection numbers for divisors on surfaces and give a representative divisor for an invertible sheaf, which includes computing canonical divisors. The functionality in parts relies on sheaf operations or slight adaptations and so requires the base scheme to be ordinary projective.
- An intrinsic has been added to test whether an isolated singularity on a scheme is analytically isomorphic to a hypersurface singularity, `IsHypersurfaceSingularity`. In the affirmative case this will also return the analytically equivalent hypersurface equation expanded to desired precision and give the transformation from the original coordinates. The user can further expand the analytic hypersurface equation at a later stage with `HypersurfaceSingularityExpandFurther` and also expand a rational function in the local analytic coordinates with `HypersurfaceSingularityExpandFunction`.

6.2 Sheaves

New Features:

- `ZeroSubscheme` returns the vanishing subscheme for a global section of a locally free sheaf.
- `HorrocksMumfordBundle` returns the Horrocks-Mumford bundle on projective 4-space as a locally free sheaf of rank 2.
- `Restriction` gives the pullback of a sheaf to a subscheme.

Changes:

- `IsLocallyFree` has finally been fixed (thanks to Eric Rains for pointing out the bug). It has been re-implemented using Fitting ideals with an adaptation of the old method as an alternative. The latter now computes an *ëtale stratification* and applies the old method inductively down it.

6.3 Algebraic Curves

New Features:

- A package of functions to compute smallest degree functions/maps to $\mathbf{P}^1$ as well as smallest degree birational plane models for low genus curves has been added. These are geometrically smallest degree, meaning they may be constructed over a finite extension of the base field in some cases. The relevant intrinsics are summarised in the next four items.
- `CliffordIndexOne`. For a trigonal curve of any genus ≥ 3, computes an explicit degree 3 map to $\mathbf{P}^1$. In the exceptional Clifford index one where the curve has genus 6 and gonality 4, computes an explicit birational map to a non-singular plane quintic. This uses Lie algebras via the algorithm of Schicho and Sevilla.
- `GenusNGonalMap` - here N can be `2` to `6`. Computes an explicit gonal (smallest degree) map to $\mathbf{P}^1$ for an algebraic curve of genus 2,3,4,5 or 6. For the hyperelliptic cases, uses the existing code; for the Clifford index one cases uses the previous intrinsic and for the genus 5 and 6 gonality 4 cases uses an algorithm of Harrison.
- `Genus5PlaneCurveModel` and `Genus6PlaneCurveModel`. Computes smallest degree (5 or 6) birational plane models of genus 5 or 6 curves that are not hyperelliptic or double covers of a genus 1 curve. Uses a variant of the method used to compute gonal maps.
- The new intrinsic `IntersectionNumbers` uses an algorithm of Jan Hilmar and Chris Smyth to compute and return all intersection places along with the corresponding local intersection multiplicity for two plane projective curves over $\mathbf{Q}$ in a single computation. (To be included in an early patch release).
- The intrinsic `Completion` applied to a function field of a curve at a place of the curve can now be computed directly. (V2.18-8)

Changes:

- `Expand` can now be applied to places of degree greater than 1. (V2.18-8)

6.4 Algebraic Surfaces

New Features:

- A basic surface type `Srfc` has been added for two-dimensional geometrically integral schemes over a field.
- Invariants, covariants and contravariants of cubic surfaces are accessible via `ClebschSalmonInvariants, LinearCovariantsOfCubicSurface`, `ClassicalCovariantsOfCubicSurface, ContravariantsOfCubicSurface`. Further, the degree 100 skew-invariant of a cubic surface can be computed via `SkewInvariant100`.
- New functionality has been added for general surfaces but applying mainly to ordinary projective models with very restricted singularity. This includes the following.
- `IsNormal`, a test for normality and `HasOnlySimpleSingularities`, which tests whether the surface only has simple (A-D-E type) singularities. The latter will also give the user the singularity type (e.g. "A3") for all geometric singular points if required.
- For ordinary projective surfaces which have only simple singularities (or are at least Gorenstein for some of the functions), computation of basic invariants with `GeometricGenus`, `Plurigenus`, `Irregularity`, `ChernNumber` (for c_1^2 and $c2$) and `HodgeNumber` (for the individual h^i, j).
- `KodairaEnriquesType` to compute the Kodaira dimension as well as the more specific type of an ordinary projective surface in the Kodaira-Enriques classification (rational, ruled, K3 etc.). The surface must again must have at worst simple singularities.
- Computation of minimal models of ordinary projective surfaces, relying heavily on adjunction and pluricanonical maps. The intrinsics differ slightly by Kodaira dimension and type so there is `MinimalModelRationalSurface`, `MinimalModelRuledSurface`, `MinimalModelKodairaDimensionZero` etc. The Kodaira dimension one case also returns a fibration map to a smooth projective curve, presenting the minimal model as a globally minimal elliptic fibration. These intrinsics require the original surface to be non-singular except in the general type (Kodaira dimension 2) case, when simple singularities are allowed.
- For ordinary projective general type surfaces with at most simple singularities, computation of the full canonical model (in weighted projective space) or canonical coordinate ring with `CanonicalWeightedModel` and `CanonicalCoordinateIdeal`.
- Computation of random nonsingular surfaces in $\mathbf{P}^4$ from a number of the families described in the paper of Decker, Ein and Schreyer. For example, `RandomRationalSurface_d10g9` and `RandomEllipticFibration_d7g6`.

Changes:

- The code to work with (singular) hypersurfaces in $\mathbf{P}^3$ has been modified so that many of the arguments of intrinsics are now `Srfc` types rather than polynomials.
- The old `Plurigenus` intrinsic has been renamed `PlurigenusOfDesingularization`.
- The old `ClassifyProjectiveSurface` has been renamed `ClassifyRationalSurface` and returns more descriptive data.

6.5 Toric Varieties

New Features:

- Algorithms have been implemented for computing the Demazure roots of the complete fan F. The intrinsic `DemazureRoots(F)` returns the roots partitioned into two sets: the semistable roots and the unipotent roots. The number of Demazure roots can be found via `NumberOfRoots(F)`.
- The new intrinsic `IsSemistable(F)` can be used to determine whether the complete fan F is semistable. Similarly, `IsReductive(X)` returns true if and only if the projective variety X has reductive automorphism group.
- The intrinsics `IsIsomorphicToProjectiveSpace(X)` and `IsIsomorphicToProductProjectiveSpace(X)` use the Demazure roots of the associated fan to determine whether the complete toric variety X is isomorphic to a (product of) projective space.

Changes:

- The intrinsics `IsPrincipal(D)` and `IsLinearlyEquivalent(D,E)` for toric divisors D and E now also return a rational function f such that $D = div(f)$ or $D = E + div(f)$, respectively.
- A significant speed improvement and reduction in the memory footprint of `CartierToWeilMap(X)` has been made. This intrinsic is fundamental when working with toric divisors, and returns the embedding map from the lattice of torus-invariant Cartier divisors to the lattice of torus-invariant Weil divisors.

7 Arithmetic Geometry

7.1 Rational Curves and Conics

Changes and Removals:

- The obsolete (and essentially ignored) varargs `Maxtrial` and `PrimalityProof` have been removed from the `Conic(X, f)` signature.

7.2 Elliptic Curves

7.2.1 Elliptic Curves over the Rational Field

New Features:

- Given the real and imaginary periods of a minimal model for an elliptic curve over **Q**, the intrinsic `EllipticCurveFromPeriods` will return the curve.

Changes and Removals:

- The extension of John Cremona's database for elliptic curves over **Q** having conductor up to 300,000 has been installed.
- Renumbered 98 curves in the above database, as requested by John Cremona, to reflect his desired new ordering. The affected curves had conductors between 130,000 and 230,000.

Bug fixes:

- A bug has been fixed that could cause the order of a torsion point on a non-integral elliptic curve over the rationals to be erroneously reported as infinite.
- Saving a workspace when the elliptic curve database is around no longer causes the database to return read errors thereafter.
- A bug has been fixed in the rank computation for elliptic curves with no rational 2-torsion points. This bug could cause some covering quartics to be erroneously discarded as not contributing to the 2-Selmer group, which in turn could cause the upper bound on the rank to be incorrectly computed. In most instances an error message or warning would highlight that this had occurred.
- Significant improvements and bug fixes have been made in `EightDescent`.
- A problem in `TwoDescent` has been fixed: for curves of large conductor having one nontrivial 2-torsion point, computations got stuck due to bad handling of power products.

7.2.2 Elliptic Curves over Finite Fields

Bug Fixes:

- A rare crash when deleting an elliptic curve over a finite field has been fixed.

7.2.3 Elliptic Curves over Function Fields

New Features:

- The routine `EllipticCurveSearch` performs a search for elliptic curves with given conductor (or conductors in a given set) defined over $\mathbf{F}_q(t)$. The amount of effort used in the search is specified by the user. The user may also specify some traces of Frobenius, when known, which greatly speeds up the search. The algorithm is nontrivial, and the implementation is carefully tuned.
- `TraceOfFrobenius` has been added for curves over $\mathbf{F}_q(t)$.

Changes:

- For an elliptic curve over a *rational* function field F, `Conductor` now returns a sequence of places and multiplicities. Previously it returned the conductor as a divisor over a trivial extension of F.

7.3 Hyperelliptic Curves

New Features:

- The `EulerFactor` of a hyperelliptic curve over the rationals or a number field can now be computed at a prime or prime ideal.

  An intrinsic for `Specialization` of a hyperelliptic curve over a function field has been added.
- The intrinsic `Specialization` has been extended to allow the calculation of a specialisation of an hyperelliptic curve defined over certain types of function field.
- `RankBound` and `RankBounds` now provide sharper upper bounds for the Mordell-Weil rank of a hyperelliptic Jacobian over a number field in some cases. This is achieved by using additional information on the Pic^1 torsor.
- A new implementation of `TwoSelmerGroup` with improved performance particularly for higher genus Hyperelliptic Jacobians over Q has been made available.
- `RankBounds` is now available for higher genus hyperelliptic Jacobians.
- `RankBounds` and `RankBound` are now available for Jacobians of cyclic covers of the projective line.
- Descent on Jacobians of cyclic covers of the projective line can now be performed using the intrinsics `phiSelmerGroup` and `PicnDescent`.
- Descents on the Pic^1 torsor of a cyclic cover of the projective can now be performed using the intrinsics `Pic1Descent` and `PicnDescent`.
- `qCoverDescent` is now available for cyclic covers of the projective line which have a singular model.
- Whether a cyclic cover of the projective line has index one over a local field can now be determined using `HasIndexOne` and `HasIndexOneEverywhereLocally`.
- New package of functions from Lercier and Ritzenthaler for genus 3 hyperelliptic curves for invariants and curve reconstruction.
- Compute the Shioda and Maeda invariants of a genus 3 hyperelliptic curve over a field of characteristic 0 or ≥ 11 with `ShiodaInvariants` or `MaedaInvariants`.
- Compute a genus 3 curve in the correct isomorphism class for given Shioda invariants as well as the geometric automorphism group with tt HyperellipticCurveFromShiodaInvariants.

– Compute geometric automorphism groups for individual genus 3 curves or compute a list of all possible automorphism groups along with the number of $\bar{k}$ isomorphism classes over genus 3 curves over k for a given finite field k of characteristic ≥ 11.

– `Twists` has been extended to compute all twists of a genus 3 curve over a finite field of characteristic ≥ 11, using the Lercier-Ritzenthaler package.

7.4 L-Series

New Features:

– The 'eq' operator now equates L-series of elliptic curves over $\mathbf{Q}$ that are isogenous.

– The `EulerFactor` intrinsic now has an `Integral` vararg.

Bug fixes:

– The conductor of the product of two L-series returned by the intrinsic `TensorProduct` now always has type `RngIntElt`.

7.5 Brandt Modules

New Features:

– Brandt modules are now implemented for quaternion orders over $\mathbf{F}_q[t]$,

– A new implementation of Brandt modules for quaternion orders over $\mathbf{Z}$ is included.

– Brandt modules are defined using the intrinsic `BrandtModule`, which takes an order and an integer, which is the Eichler level.

– The algorithms efficiently handle the level: they do not explicitly work with the ideal classes of the Eichler order, but only those of the maximal order. (This is the same approach as in the algorithm used to compute Hilbert modular forms.)

– The dimension of a Brandt module can be computed (without creating a module) using either `BrandtModuleDimension` or `BrandtModuleDimensionOfNewSubspace`.

– The `HeckeOperator` of a Brandt module can be computed for any prime not dividing the level.

– The common eigenvectors of the Hecke operators can be obtained efficiently using `HeckeEigenvectors`. Eigenvalues at additional primes can be then obtained using `HeckeEigenvalue`.

– The types `ModBrdtNew` and `ModBrdtNewElt` which have been introduced for the new cases will revert to `ModBrdt` and `ModBrdtElt` in a future release.

7.6 Modular Forms

Changes and Removals:

– `OverconvergentHeckeSeries` has been sped up, by a factor of more than 10 in many cases. The `DegreeBound` parameter has been removed (it is no longer useful with the improved code).

7.7 Modular Symbols

Bug fixes:

– Bugs have been fixed in `IsTwist` and `IsMinimalTwist`. In addition, the proper bounds are now used, making the functions much faster.

7.8 Admissible Representations

Bug fixes:

– Several bugs and minor glitches have been fixed in this package.

8 Arithmetic Fields (Global)

8.1 Algebraic Number Fields

New Features:

- A new implementation of the standard (`NoSieve`) algorithm for `ClassGroup` of general fields is now used. Fields of larger degree are handled much better by the new implementation. For example, (conditional) class groups and units can be obtained for the degree 24 fields arising in 5-descent on elliptic curves, for a reasonable range of conductors. In addition, running times are much more stable than before. There will be further improvements, leading to significant gains in speed. (The default choice of algorithm, `Sieve` or `NoSieve`, has not changed: `Sieve` is used by default for fields of degree at most 5 with large discriminant. The algorithm may be selected by the user, in all cases except that `Sieve` is not allowed for very small discriminants.)
- `ClassGroupExpectedNumberOfTrials` estimates the difficulty of finding relations in the standard class group algorithm for a given factor base bound. Based on this, `ClassGroupSuggestedBound` gives a suitable value for the factor base bound.
- `Automorphisms` can now be applied to a number field that is represented as an extension of another number field.

Changes and Removals:

- The computation of coercions between orders and fields has been made more efficient. Previously when an order or field was constructed all possible coercion paths were computed and stored. In some cases this caused the graph which stored this information to be excessively large and cause Magma to use an excessive amount of memory. In all cases there was the time cost of computing coercion paths which may not have been used. Both of these problems have been addressed by computing and storing coercion paths only when they are required. Examples which ran into problems with memory usage no longer have these problems, other examples have seen significant speed-ups.
- The choice of algorithm used to compute `Subfields` of simple extensions of $\mathbf{Q}$ has been improved (V2.18-3, V2.18-8).
- Improvements have been made to the implementation of the Klüners-van Hoeij-Novocin algorithm for the computation of `Subfields` of simple extensions of $\mathbf{Q}$. These improvements were made to the LLL computation, the prime selection and the computation of the subfield polynomial. Some of these improvements were patched in V2.18-8.
- When applied to an element of a number field or a field of fractions of an order of a number field `IsIntegral` now returns (as a second return value) a denominator such that the denominator times the input is integral.
- `CoveringStructure` is now provided for rings of straight line polynomials which are used in the computation of Galois groups of polynomials over global arithmetic fields.
- A straight line polynomial (as used in the computation of Galois groups of polynomials over global arithmetic fields) can now be evaluated using a coefficient ring map which maps the coefficients of the straight line polynomial into the universe of the point at which the polynomial is being evaluated.

Bug Fixes:

– A number of bugs have been fixed in the `Sieve` algorithm for `ClassGroup`. In addition, problems with the computation of S-units in cases which used the `Sieve` algorithm have been corrected.

– A bug with incompatible sequence elements in `SplittingField` has been fixed. Thanks to A. Elkin for the correction.

– Compatibility of modules over maximal orders has been fixed.

– The `PrimitiveElement` of an order is now integral.

8.2 Characters and Artin Representations

New Features:

– Intrinsics `CentralCharacter` have been added for Dirichlet and Hecke characters (including Grössencharacters). Similarly with `Determinant` for Artin representations.

– The intrinsics `DirichletCharacter` and `HeckeCharacter` can now take torsion units from a number field as their input images.

– Utility intrinsics `DirichletCharacterOverNF` and `DirichletCharacterOverQ` have been added to allow one to pass between the different types of Dirichlet characters.

– Intrinsics `TargetRestriction` have been added for Dirichlet and Hecke characters.

– Deriving an `ArtinRepresentation` from a Dirichlet character has been improved, via superior computation of the Dirichlet kernel class field.

8.3 Algebraic Function Fields

New Features:

– The intrinsics `MaximalOrderFinite` and `MaximalOrderInfinite` now apply to abelian extensions of function fields. These intrinsics can be much faster than the same intrinsics applied directly to the function field of the abelian extension.

– When computing the `GaloisGroup` of a polynomial over a function field having characteristic 2, two more invariants are now available, similarly to the odd characteristic cases. One of these invariants is used to decide whether the Galois group is a subgroup of A_n.

– When computing the `GaloisGroup` of a polynomial over a function field having prime characteristic, invariants with coefficients in $F_q[t]$ are now available.

– A function field may now be constructed from a sequence of multivariate polynomials. The resulting field will have a similar representation to that constructed from one multivariate polynomial.

– `Embed` can now be applied to function fields represented as an extension by multiple defining polynomials.

Changes and Removals:

– Improvements have been made to the computation of maximal orders of Artin–Schreier extensions. The application of this more efficient algorithm is restricted to fields whose constant field is perfect as these are the fields for which the algorithm is known to always work.

– When computing the `GaloisGroup` of a reducible polynomial over a function field of prime characteristic the descent step has been split so that it is only performed for those factors whose splitting fields may have non-trivial intersection.

– The order of the variables (when there are at least two) in `RationalFunction` has been reversed back to what it was in Magma V2.10. Now the first variable corresponds to the primitive element of the topmost extension.

Bug Fixes:

– Compatibility of modules over maximal orders has been fixed.

– The `PrimitiveElement` of an order is now integral.

9 Arithmetic Fields (Local)

9.1 p-adic Rings and their Extensions

New Features :

– Roots of polynomials over general local fields can now be computed.

– The intrinsic `CoveringStructure` may now be applied to a general local field together with a compatible field.

Bug Fixes:

– The ramified representation mapping has been fixed for some general local fields (V2.18-10).

– A typo has been fixed in the Magma-level printing of extensions of a p-adic field; this error would stop the result from being able to be properly `eval`'ed.

– A memory leak has been fixed in the p-adic coercion for large primes p.

– Some issues with p-adic ring and field creation have been fixed; those could have caused two equal p-adic structures to be erroneously considered different, causing coercion problems.

– Asking for the defining polynomial of a local ring with respect to itself no longer crashes.

9.2 Series Rings

Bug Fixes:

– A bug has been fixed that could cause loss of precision when one power series was evaluated at another.

10 Basic Rings and Fields

10.1 Integer Ring

New Features:

– Maps between **Z** and a residue class ring or field can now be applied directly to matrices over **Z**.

Bug Fixes:

– A crash when calling `Binomial` or `NumberOfPermutations` on large integers has been fixed.

– A bug which caused binomials to be computed in a very inefficient way in some cases has been fixed.

10.2 Finite Fields

New Features:

– This release includes a major new implementation by Allan Steel of the Coppersmith index-calculus algorithm for computing discrete logarithms of finite fields of small characteristic (for characteristic greater than 2, a straightforward generalization of Coppersmith's algorithm is used). This means that discrete logarithms can now be quickly computed for many fields of small characteristic for the first time.

A suite of auxiliary tables boost the algorithm so that the initial computation of logarithms of a factor base can be avoided. This means that logarithms of individual elements can be computed immediately if a relevant table is present for the specific field. By default, tables are included in the standard Magma distribution at least for all fields of characteristic 2, 3, 5 or 7 with cardinality up to 2^{200}.

The user can optionally download a much larger suite of tables from the Magma optional downloads page `http://magma.maths.usyd.edu.au/magma/download/db/` (files `FldFinLog_2.tar.gz`, etc.; about 5GB total). The complete suite includes tables for the fields $\mathrm{GF}(p^d)$ for:

- $p = 2$: all $d \leq 440$;
- $p = 3$: all $d \leq 261$;
- $p = 5$: all $d \leq 203$;
- $p = 7$: all $d \leq 150$;
- $p = 11$: all $d \leq 85$;
- $p = 13$: all $d \leq 72$;
- $13 < p < 97$: several non-trivial fields.

– The function `PrimitiveElement(K)` has been changed so that if the default generator of a finite field K is not primitive, then the primitive element of K is taken to be the element of K corresponding to the first polynomial over the base field of K in lexicographical order which is irreducible (previously, a random primitive element of K was chosen). This means that the same fixed primitive element will now always be chosen for a fixed extension of a prime field (and also implies consistency for the base with respect to which discrete logarithms are computed by default).

Changes and Removals:

– The procedure `Coppersmith` has been removed, since it is not needed now.

10.3 Nearfields

New Features:

- This release includes a new package implementing finite nearfields and related finite projective (non-Desarguesian) projective planes.

  A nearfield satisfies all the axioms of a field except for the commutative law of multiplication and one of the distributive laws. In the Magma implementation the nearfields satisfy the right distributive law.

  - The finite sharply doubly transitive permutation groups are in one-to-one correspondence with the finite nearfields.
  - Nearfields coordinatise a class of translation planes.
  - Nearfields and they are the starting point for the construction of the Hughes planes.
- The function `DicksonNearfield(q,v)` creates a Dickson nearfield from the Dickson pair (q, v), and the function `ZassenhausNearfield(n)` create one of the 7 exceptional nearfields.
- The group of units of a nearfield is metacyclic: it is available as a matrix group, a permutation group and as PC-group.
- The function `AffineGroup(N)` returns the semidirect product of the additive group of N by the group of units. All sharply doubly transitive groups occur in this form.
- Nearfields can be tested for isomorphism and the full automorphism group of a nearfield has been implemented.
- The function `ProjectivePlane(N)` returns the non-Desarguesian projective plane coordinatised by the nearfield N.
- The function `HughesPlane(N)` returns the Hughes plane obtained from N. Note that N can be either a Dickson nearfield or a Zassenhaus nearfield. (In the literature, the planes obtained from Zassenhaus nearfields are often called generalised Hughes planes.)

10.4 Real and Complex Fields

Bug fixes:

- A crash related to the new garbage collector and the extended reals has been fixed.

10.5 Polynomial Rings

New Features:

- A quotient of two polynomials can now be coerced into the quotient of an appropriate polynomial ring.

11 Coding Theory

11.1 General Linear Codes

Bug Fixes:

- A bug in `MinimumWords` has been fixed; this bug could cause it to return spurious words not in the code when called on some quasicyclic codes.
- A rare minor leak in the minimum weight algorithm has been fixed.

11.2 Linear Codes over Finite Fields

New Features:

- Given a polytope P, or sequence of points S, the corresponding (generalised) toric code can be calculated using `ToricCode`.

Bug Fixes:

- For linear codes over non-binary finite fields, under some circumstances the automorphism group of the dual was being incorrectly used as the automorphism group of the code. This has been fixed.
- Additionally, for such codes a reversed test in the automorphism group computation would cause this computation to be performed over a less efficient choice of code and dual. This has also been fixed.
- Calling `BCHBound` on a code with generator polynomial $(x^n - 1)/(x - 1)$ now correctly returns a bound of n instead of 1.

11.3 Linear Codes over Finite Rings

Bug Fixes:

- A bug in `StandardForm` applied to $\mathbf{Z}_4$ codes has been fixed; this bug would cause the result to not always have the required diagonal property.
- A bug in `ParityCheckMatrix` applied to $\mathbf{Z}_4$ codes has been fixed; this bug could cause the parent of the returned matrix to have the wrong number of rows.

11.4 Quantum Error-Correcting Codes

Bug Fixes:

- A bug has been fixed which would previously cause the components of the weight distribution of a self-dual quantum code to be all set to zero upon creation if the weight distribution of the stabilizer code was already known.

12 Combinatorial Theory

12.1 Graphs

Changes and Removals:

- The directed graph constructors have been extended to allow edges to be given as sequences in addition to sets (as was formerly the case).
- Spanning trees now retain vertex labels of the containing graph (edge labels will be lost).

Bug Fixes:

- Assigning an explicitly empty sequence of edge labels to a graph and then asking for the edge labels no longer causes a crash.

12.2 Matrices

New Features:

- A version of the intrinsic `AutomorphismGroup` which computes the group of row and column permutations of a matrix that fix the matrix has been installed. The corresponding `IsIsomorphic` intrinsic is also available.

13 Commutative Algebra

13.1 Polynomial Rings

Bug Fixes:

- A crash in `DistinctDegreeFactorization` when the `Degree` parameter was specified has been fixed.

13.2 Ideal Theory and Gröbner Bases

New Features:

- The algorithm for computing a Gröbner basis over a euclidean ring has been greatly sped up (by more intelligent pair selection).
- The algorithm for computing a Gröbner basis over $\mathbf{Z}$ has been greatly sped up by using modular techniques where possible.
- The algorithm for computing the primary decomposition of a zero-dimensional ideal over $\mathbf{Q}$ been greatly sped up for several classes of inputs.

14 Geometry

14.1 Convex Polytopes and Polyhedra

A number of new features have been added by Al Kasprzyk to his polytopes package.

New Features:

- A native version of PALP normal form has been implemented, and is available via the intrinsics `PALPNormalForm(P)` and `NormalForm(P)`. This is the first implementation of the famous algorithm by Max Kreuzer and Harald Skarke outside of PALP. An alternative algorithm has also been developed which is significantly faster when the polytope P has symmetries.
- `AffineNormalForm(P)` can be used to obtain the affine normal form of the maximum dimensional lattice polytope P.
- Minkowski decompositions of polytopes in arbitrary dimensions can now be computed. This is based on a result of Klaus Altmann.
- The Newton polytope of a rational function f (regarded as a Laurent polynomial) can be calculated via `NewtonPolytope(f)`.
- The new `PointProcess(P)` and `PointProcess(C)` intrinsics can be used to iterate over the points in a polytope P and a pointed cone C.
- Given a primitive form v (i.e. a point in the lattice dual to L), `ChangeBasis(v)` returns a change of basis of L such that the kernel of v is mapped to the standard codimension one lattice.
- A database of all (2-dimensional) facets of the canonical Fano 3-dimensional polytopes has been added. These polygons, which can be accessed via `PolytopeCanonicalFanoDim3Facet`, are defined up to affine equivalence.
- A database of small polygons has been implemented. Once installed, this can be accessed via `PolytopeSmallPolygon`.
- Reverse database look-up has been implemented in three-dimensions. Given a polytope P, the intrinsic `DatabaseID(P)` will return a sequence of matching entries in the databases of three-dimensional polytopes. In the case when P is a canonical Fano polytope, the latest version of the `canonical3` database must be installed in order to access reverse look-up data.

Changes:

- The intrinsic `IsIntegrallyClosed(P)` can be used to determine whether a polytope P is integrally closed (i.e. every lattice point in kP can be written as the sum of k lattice points in P, for all $k \in \mathbf{Z}$). The algorithm has been significantly improved, leading to much earlier detection of a negative answer.
- New special cases have been added to point enumeration for a polytope P, significantly improving the time taken under certain extreme choices of basis (which, unfortunately, includes some presentations of reflexive simplices that have become standard).

14.2 Finite Geometry

As a consequence of the recent implementation of finite near-fields, constructions for finite planes belonging to families whose definition is based on near-fields have been implemented.

New Features:

- The function `ProjectivePlane(N)` returns the non-Desarguesian projective plane coordinatised by the nearfield N.
- The function `HughesPlane(N)` returns the Hughes plane obtained from the near-field N. Note that N can be either a Dickson nearfield or a Zassenhaus nearfield. (In the literature, the planes obtained from Zassenhaus nearfields are often called generalised Hughes planes.)

14.3 Incidence Geometry

New Features:

- The intrinsic `LocallySArcTransitive` has been implemented by Dimitri Leemans for coset geometries. It returns the largest integer s such that cosert geometry D is locally s-arc-transitive but not locally (s+1)-arc-transitive.

15 Groups

15.1 Automatic Groups

New Features:

- A number of `intrinsics` have been provided to return information about the word acceptor and word difference automata. These are `WordAcceptorSize`, `WordAcceptor`, `WordDifferenceSize`, `WordDifferenceAutomaton`, `WordDifferences`, and `GeneratorOrder`.
- Printing level Magma has now been implemented for automatic groups. This makes it possible to write (read) automatic groups to (from) a file. The reconstruction of the group implements a number of basic checks on the input to avoid possibly corrupt input, but does not do exhaustive checks.
- A database of automatic groups corresponding to 4000 hyperbolic 3-manifolds from the Weeks' census of low volume hyperbolic 3-manifolds will shortly be made available.

Bug Fixes:

- Detection of infinite loops when constructing an automatic structure for a group has been improved. Also, the verbose messages printed on failure are more comprehensive.
- A number of memory problems, including memory leaks and crashes, have been fixed, improving the use of memory and the stability of the code.
- The generator access `A.i`, when `A` is an automatic group, now works when `i` is zero or negative.
- The `MaxWordDiffs` parameter is now taken notice of when constructing the automata.

15.2 Classical Groups

Bug Fixes:

- A number of errors in the maximal subgroups of the classical groups of Lie type having degree up to 12 have been fixed. A definitive list of these groups are being prepared for publication by John Bray, Derek Holt and Colva Roney-Dougal. Parallel to this is a suite of Magma routines developed by Derek and Colva which are designed to return the maximal subgroups of any group of this type.

15.3 Finitely Presented Groups

New Features:

- Given a 2-generator finitely presented group G, the version of the Plesken-Fabianska algorithm developed by Sebastin Jambor simultaneously computes all quotients of G which are isomorphic to some $\mathrm{PSL}(2, q)$, for any prime power q.
- The intrinsic `L2Quotients`(G) returns a list of prime ideals of $Z[x_1, x_2, x_{12}]$ which contains all information about the L_2-quotients.
- The intrinsic `L2Type`(P) applied to any prime ideal P in $Z[x_1, x_2, x_{12}]$, returns a string describing the type of L_2-quotient encoded by the ideal.
- The intrinsics `L2Generators`(P) and `L2Ideals`(I) return further information about the ideals returned by `L2Quotients`.

15.4 Matrix Groups

New Features:

- The `SubgroupLattice` intrinsic has been extended to work with finite matrix groups.

Changes:

- When constructing a matrix group, the inverses of the generators are no longer computed immediately. Inverses are computed only when required.
- The algorithms used for computing the orbit of a matrix group on the natural 1-dimensional points have been changed to use faster methods where possible.
- The `OrbitImage` intrinsic now returns both the permutation group and the orbit.
- Steps have been taken to avoid constructing a base and strong generating set whenever possible.

15.5 Matrix Groups Over Associative Algebras

Changes:

- Matrix groups over associative algebras have been revised slightly to conform more fully with the general matrix group functionality.

15.6 Matrix Groups Over Finite Fields

The following notes describe new features that have been developed for use in the context of the Composition Tree representation of a matrix group.

New Features:

- Short presentations on standard generators for the orthogonal groups have been installed. The availability of presentations enables rapid verification of the Composition Tree for any group whose composition factors exclude exceptional groups of Lie type.
- The intrinsic `ClassicalRewrite` writes an element of a classical group G as an SLP in the standard generators of G.
- Lifting-style algorithms have been implemented by Derek Holt for further types of structural calculations in matrix groups using the Composition Tree data structure. These include `LMGCentraliser`, `LMGIsConjugate` (for elements), `LMGConjugacyClasses`, `LMGNormaliser`, `LMGIsConjugate` (for subgroups) and `LMGMaximalSubgroups`. In the near future, each intrinsic prefixed with the letters "LMG" will be merged with the corresponding standard intrinsic.
- A new intrinsic `AffineGroup` takes a matrix group G over a finite field and returns the semidirect product of G by the natural G-module as a permutation group where the degree is the size of the natural module.

15.7 Matrix Groups Over Z and Q

The intrinsics described in this section have been implemented by Marcus Kirschmer (Aachen) unless noted otherwise.

New Features:

- Given a matrix A of finite order in $\mathrm{GL}(n, \mathbf{Z})$, a much improved algorithm is used to implement the intrinsic `CentralizerGLZ`(A) which computes the centralizer of A in $\mathrm{GL}(n, \mathbf{Z})$.
- The intrinsic `CentralizerGLZ`(A)) may now be used to compute the centraliser of a matrix A in $\mathrm{GL}(2, \mathbf{Z})$, when A has infinite order. The algorithm and code was developed by D. Husert (University of Paderborn).
- A new intrinsic `IsGLZConjugate`(A, B) tests whether two rational or integral matrices A and B having finite order are conjugate in $\mathrm{GL}(n, \mathbf{Z})$ or $\mathrm{SL}(n, \mathbf{Z})$.
- The intrinsic `IsGLZConjugate`(A, B) may be used to test conjugacy of matrices A and B in $\mathrm{GL}(2, \mathbf{Z})$, when A and B have infinite order. The method and code are due to D. Husert.
- Given a finite subgroup G of $\mathrm{GL}(n, \mathbf{Z})$, a much improved algorithm is used to implement the intrinsic `NormalizerGLZ`(G) which computes the normalizer of G in $\mathrm{GL}(n, \mathbf{Z})$.

15.8 Permutation Groups

New Features:

- It is now possible to identify a transitive permutation group of degree 32 in the database of all transitive groups of degree 32; this requires the installation of an optional database containing identification data.
- A new intrinsic `AffineGroup` takes a matrix group G over a finite field and returns the semidirect product of G by the natural G-module as a permutation group where the degree is the size of the natural module.
- A new intrinsic `pCoreQuotient` has been added. It returns the quotient of the given permutation group G by it's p-core as a permutation group.

Changes:

- A number of measures have been taken to avoid constructing base and strong generating set. Notably for the image of a homomorphism, when testing a permutation group for being alternating or symmetric, and after a backtrack search.

15.9 Quasisimple Groups

Quasisimple Groups

- A table of modular representations of all quasisimple groups up to degree 100 can be accessed using the intrinsic `QuasisimpleMatrixGroup`. The list of names of the available quasisimple groups can be obtained using the intrinsic `QuasisimpleMatrixGroups`.

15.10 Soluble Groups

New Features:

- A package for the more efficient calculation of the automorphism group of a finite soluble group G is included. The algorithm was developed by David Howden and works by extending the automorphisms of a large Sylow p-subgroup of G to the automorphism group of G. A slight variation of the algorithm may be used to test isomorphism. The relevant intrinsics are `AutomorphismGroupSolubleGroup` and `IsIsomorphicSolubleGroup`.
- Intrinsics `OrbitStabilizer` for computing orbit and stabilizer of an element under the action of a finite soluble group has been added. The intrinsic `IsConjugate` is extended to take advantage of the data computed by `OrbitStabilizer`.

16 Lattices

16.1 Lattices

New Features:

- The intrinsic `AutomorphismGroup`(L, F) has been extended to compute the subgroup of the automorphism group of a lattice which fixes a collection of forms setwise. The forms are not required to be positive definite or even symmetric. This is a variation of the partition-refinement backtrack lattice automorphism program developed by Bill Unger in 2010 so it is fast. The new algorithm also applies to computing an isometry between two lattices which carries a set of forms attached to one lattice to a set attached to the second.
- The intrinsic `IsRationallyEquivalent` tests whether a pair of lattices are rationally equivalent.

Changes and Removals:

- The `WittInvariants` intrinsic now has a `Minimize` vararg.

Bug Fixes:

- The `pSignatures` and `pExcesses` intrinsics now return an empty sequence when the input has determinant zero.
- A bug when computing isometries of lattices with large entries in the Gram matrix has been fixed.
- A memory leak when computing automorphisms and isometries of lattices has been fixed.

17 Lie Theory

17.1 Lie Algebras

New Features:

- The basic multiplication algorithm for elements of an algebra has been sped up.
- The creation of subalgebras and quotient algebras has been greatly sped up in general.
- The algorithm to compute a composition series has been greatly sped up in general.
- The new intrinsic `CompositionFactors` returns a sequence containing the composition factors of a Lie algebra defined by structure constants. In fact, the value returned by this intrinsic is the same as the second value returned by `CompositionSeries`, but `CompositionFactors` can be very much faster than the latter function.
- The intrinsic `PleskenLieAlgebra` constructs the Plesken Lie algebra of a finite group G. This is the linear span of the elements $g - g^{-1}$ in the group algebra of G.

17.2 Representation Theory

New Features:

- The W-graph machinery has been extended to support directed W-graphs. The code was supplied by Bob Howlett.

18 Linear Algebra and Module Theory

18.1 Matrices

New Features:

- The intrinsic function `Evaluate(f, A)`, where f is a univariate polynomial and A is a matrix over a finite field, has been greatly sped up, particularly in the case that f has high degree.
- A new intrinsic function `Evaluate(f, S)` has been added, where f is a univariate polynomial and S is a sequence of matrices; the function returns the evaluation of f of every entry of S as a sequence. This function will often be faster than evaluating f at each element of S separately (at least when the matrices over a finite field for the moment).
- The Hermite Normal Form algorithm has been greatly sped up for sparse matrices of full rank which have at least one large elementary divisor (this benefits algorithms such as the index calculus class group algorithm).
- Better handling has been introduced for matrix algorithms over residue class rings in the case where it is not easy to determine quickly whether such rings are fields.

18.2 Sparse Matrices

New Features:

- The Lanczos algorithm (selected by the `Lanczos` parameter to the `ModularSolution` function) has been greatly sped up.

18.3 Bilinear and Sesquilinear Forms

New Features:

- A polar space, namely a vector space with an attached bilinear, sesquilinear or quadratic form, can be constructed from a standard form or from a user-supplied form. The type of a polar space is one of: symplectic, pseudo-symplectic, unitary, quadratic or orthogonal, according to whether the form is alternating, pseudo-alternating, hermitian, quadratic or symmetric.
- New functions `IsometryGroup(V)` and `SimilarityGroup(V)` return the group of all isometries and the group of all similarities of the polar space `V`.
- Given polar spaces `V` and `W`, the function `IsIsometric(V,W)` determines whether there is an isometry from `V` to `W` and, if so, returns it.
- The function `PseudoSymplecticGroup(n,q)` returns the pseudo-symplectic group of $n \times n$ matrices over the field GF(q) (where q must be a power of 2).
- The function `LieAlgebraFromForm(J)` returns the Lie algebra of derivations of the form `J`.
- The functions `HyperbolicPair`, `Witt Decomposition` and `ExtendIsometry` have been enhanced to work with all types of polar spaces. In particular, Witt's Theorem (using `ExtendIsometry`) is now available for unitary spaces.
- Given a matrix group G there are new functions to compute the bilinear, sesquilinear and quadratic forms which are invariant (or invariant up to a scalar multiple) under the action of G. If G (and its derived group) is absolutely irreducible this functionality has previously been available via `ClassicalForms(G)`.

19 Linear Associative Algebras

19.1 Associative Algebras

New Features:

- The process of creating subalgebras and quotient algebras has been greatly sped up in general.
- A new intrinsic `CompositionFactors` is provided. This returns the same object as the second return value of `CompositionSeries`, but may be very much faster than the latter function.

Changes and Removals:

- The norm of an ideal of an order whose coefficient ring is not an order of a number field is now accessible through an attribute.
- A conjugate of an ideal will now have left and/or right orders and norm set on computation if the appropriate information is known on the input ideal.
- An ideal may now have less basis elements than the dimension of the algebra.

Bug Fixes:

- A sequence is now checked for having elements in the coefficient ring (rather than its field of fractions) before it is coerced into an order.
- The construction of an ideal whose basis has normal form with zero rows has been fixed.

19.2 Basic Algebras

New Features:

- A homomorphism of basic algebras A and B can be constructed using the hom-constructor:

  $$\texttt{hom} < \texttt{A} - > \texttt{B} | \texttt{S} >$$

  where S ia an appropriate matrix. The intrinsic `IsAlgebraHomomorphism`(A, B, ψ) can be used to verify the homomorphsim property while the intrinsic `Image` can be used to compute the image of a homomorphism.
- The intrinsic `Restriction`(M, B, ξ) constructs the restriction map for module M along the algebra homomorphism ξ.
- The intrinsic `AutomorphismGroup`(A) will compute the automorphism group of the basic algebra A. Isomorphism of basic algebras A and B can be tested using the intrinsic `IsIsomorphic`(A, B). The implementation generalizes an algorithm of Bettina Eick for nilpotent structure constant algebras. Related intrinsics are `GradedAutomorphismGroup` and `IsGradedIsomorphic`.
- Ideals, subalgebras and quotient algebras may now be constructed using the standard constructors `ideal, sub` and `quo`. The subalgebra is returned as a basic algebra together with the inclusion homomorphism. The returned subalgebra includes the minimal idempotent that act as an identity on the given elements. The intrinsic `Annihilator`(A, S) constructs the annihilator of the ideal generated by Elements S in the basic algebra A.

– The intrinsics `Centre` and `Centralizer` may now be used to construct the centre of a basic algebra and the centralizer of a collection of elements in a basic algebra.

– The intrinsic `AssociatedGradedAlgebra`(A) constructs the associated graded algebra of a basic algebra.

– Other new functions allow the user to create the basic algebra of the action algebra on the direct sum of a sequence of modules over a matrix algebra or group algebra. An example is the basic algebra of a block of a group algebra.

19.3 Matrix Algebras

New Features:

– For a matrix algebra defined over a finite field and having a unit element, faster algorithms for computing a basis, a basis of the Jacobson radical and the unit group have been added. The algorithms are due to Jon Carlson.

19.4 Quaternion Algebras

New Features:

– The intrinsic `Normalizer`(S) calculates the normalizer of an order in a definite quaternion algebra A over a field F where F is the rationals, $\mathbf{F}_q(t)$ or a number field.

– A much more efficient algorithm is now used for enumerating 2-sided ideal classes by means of the intrinsic `TwoSidedIdealClasses`(S), where S is an order in a quaternion algebra.

– The intrinsic `MaximalRightIdeals`(O, p) computes the integral ideals of norm p with left or right order O.