Summary of New Features in Magma V2.21-4

May 2014

1 Introduction

This document provides a terse summary of the new features released as part of Magma versions V2.21-1 (December 2014) and V2.21-4 (May 2015).

A small number of new features were exported in patch releases prior to the main release of V2.21 in December 2014 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 V2.20-x.

Recent releases of Magma were: V2.20 (December 2013), V2.19 (December 2012), V2.18 (December 2011), V2.17 (December 2010), V2.16 (November 2009), V2.15 (December 2008), V2.14 (October 2007).

2 Highlights

Algebraic Geometry

- *Schemes*
  - The corank 2 family of $X_{k,r}$ singularities have been added to the package for Arnold normal forms of isolated analytic hypersurface singularities.
  - Corank 3 families have been added to the Arnold normal forms package. Not all families have been included in the initial release but the corank 3 package should be completed for a patch release early next year.
  - Affine patches have been added for weighted projective spaces and patch indices where the grading is greater than 1. These are not generally plain affine ambients and do not fit directly into the existing framework for standard affine patches.
  - The tests for singularity have been extended to give correct results over the entirety of a subscheme in weighted projective space.
- *Surfaces*
  - Desingularisation by local blow-up – an alternative to formal desingularisation – has been added for surfaces. It is faster than formal desingularisation in a number of cases, rendering some computations, which would hang previously, now feasible in Magma. It is only available for surfaces with point singularities but it can be applied to such surfaces in any ambient and in any characteristic.

– Blow-up desingularisation provides additional information, not previously available with formal desingularisation. There are new intrinsics to compute the intersection matrix of divisors lying above a singular point in the desingularisation as well as computing multiplicities of these blow-up divisors in pullbacks of divisors and differentials of the singular surface. Intersections pairings of strict transforms of divisors with blow-up divisors or the canonical divisor of the desingularisation can also be computed.
– Existing functionality for computing birational invariants of desingularisations (geometric and arithmetic genus, plurigenera) and general adjoint spaces has been extended to work with the new blow-up desingularisation in addition to the formal desingularisation.
– The intrinsics for classifying a rational surface, parametrizing a rational surface and testing whether a surface is rational have been extended to work with blow-up desingularisations.

Arithmetic Geometry

- *Elliptic Curves Over* $\mathbf{Q}$ *and Number Fields*

  - The main functions for computing the Mordell-Weil group, rank, etc, for the group of rational points on an elliptic curve over $\mathbf{Q}$ or a number field have undergone a major internal change. These functions now make use of the full power of the machinery in Magma: all the descent methods, and other (analytic) methods, where appropriate. The function `MordellWeilShaInformation` does the same, but also prints information along the way, and also returns information about the Tate-Shafarevich group. There are some syntax changes, which are backwards compatible: current usage of these functions will continue to work just as well.
  - An algorithm for computing a rank bound for a curve over $\mathbf{Q}$ with a 2-isogeny, by performing several higher descent steps, has been contributed by Tom Fisher. A summary table is printed after each step, which displays the rank bounds obtained so far.
  - An implementation of 5-descent has been contributed by Tom Fisher.

- *L-Series*

  - A number of new features have been added to L-series package. The most notable of these are: Hodge structure functionality, symmetrization of an L-series, and improved computational techniques for checking functional equations. The symmetrization capability includes both orthogonal and symplectic versions where applicable. The machinery is capable of handling examples where the degree of the resulting L-function is as large as 10.
  - The evaluation (checking) of a functional equation can be expedited by the use of the tools for motivic L-series in the case of many types of L-series. This has been used in conjunction with the hypergeometric motives database that is mentioned below.
  - User utilities to change specific Euler factors and copy coefficients between L-functions have been added.

- *Local Galois Representations*

  - A new user type for Galois representations, implemented by T. Dokchitser, provides a uniform method for working with Galois representations over p-adic fields. The objects from which these can be constructed include: elliptic curves (including over number fields), Dirichlet characters, modular forms, Artin representations, and admissible representations of $\mathbf{GL}_2(\mathbf{Q}_p)$. One prospective application of this is to the computation of local information for tensor product L-functions.

- *Hypergeometric Motives*
  - A new type LSerMot has been introduced that expedites certain types of L-series computations.
  - In particular, the amount of (internal) precision required when invoking weighting functions (incomplete Mellin transforms) is now computed dynamically.
  - The principal gain is when checking the functional equation, which for L-functions of higher degree or weight is now faster and more robust.
  - A database of wild prime data for $t = 1$ degenerations of all 1582 data of degree up through 6 is now included in Magma.
  - User utilities to save Euler factors have been added, and additional methods for guessing wild Euler factors due to F. Rodriguez Villegas and D. Roberts are provided.

Arithmetic Fields

- *Class Groups of Number Fields*
  - The new class group algorithm, the first version of which was released in V2.20, is now mostly in place. Fast code has been written for each of the core subroutines. The most important subroutines are: finding short bases of ideals, a special algorithm for calculating norms, and batch smoothness testing. As an indication of the current capability, fields of degree 24 and discriminant up to 105 digits (arising in 5-descent for curves in the Cremona table) require on the order of one day. The routine is well optimized for fields of this kind, as well as for small degree. It can also handle all the cyclotomic fields that have degree 72, although it is currently far from optimized for such large degrees.
- *Galois Groups*
  - The algorithm has been improved at several points. Rational degree 20 polynomials with moderate coefficient size will usually be treated in less than one second. The internal construction of invariants is now tested for all irreducible polynomials up to degree 47. Many larger cases benefit from the improvements as well.

Basic Rings and Fields

- *Polynomial Rings*
  - The algorithm to factorise a polynomial $f \in K[x]$ where K is a number field has been significantly improved, leading to general speedups. In particular, there is often a large speedup when f has small degree and K has large degree, or f has large degree and K has small degree.
  - The algorithm to compute the roots of a polynomial $f \in K[x]$ where K is a number field has been been sped up (beyond the improvements to general factorisation above).
  - The modular algorithm for univariate polynomial GCD in $K[x]$ where K is a number field, has greatly improved (particularly for the case where the leading coefficients of the input polynomials are large).
  - The modular resultant algorithm for multivariate polynomials over $\mathbf{Z}$ or $\mathbf{Q}$ has been significantly improved.
  - The multivariate resultant algorithm has been greatly sped up by use of Bezout matrices (instead of the Sylvester matrices) when there is a larger number of terms but the degrees of the input polynomials in the main variable are quite small.

Commutative Algebra

- *Gröbner Bases*

  - The dense F_4 algorithm has been greatly sped up in general for several classes types of ideals over finite fields where there is a large amount of reduction to zero. For the Minrank Challenge C, the `grevlex` GB is computed about 2.3 times faster than when using the dense F_4 algorithm in V2.20.
  - The dense variant of the F_4 algorithm now allows a larger number of variables and higher degree than previously.
  - The general F_4 algorithm has been sped up on Intel/AMD processors with AVX support (using special AVX instructions) for ideals defined over the fields $\mathrm{GF}(2^k)$ for $k > 1$.
  - A new algorithm for interreduction of a set/sequence of polynomials has been developed, based on the Faugere F_4 algorithm. This algorithm is dramatically faster than the default algorithm for many inputs.
  - The FGLM basis change algorithm for generic zero-dimensional ideals has been greatly sped up. The Keller-Gehrig algorithm (see below) can also be used within this algorithm.
  - A major speedup has been introduced into the linear algebra phase of the F_4 algorithm for computing Gröbner bases. For example, over a small-prime finite field, the speedup for computing the Cyclic-10 GB is about a factor of 4 and the speedup for computing the Katsura-12 GB is about a factor of 20. Currently the improvement only applies to ideals over small-prime finite fields, but it is hoped that it will be extended to other kinds of fields soon. (V2.21-4)
  - A major new asymptotically-fast modular algorithm has been developed for computing Gröbner Bases of ideals defined over algebraic number fields. The new modular algorithm handles fields of arbitrary degree (while the previous modular algorithm was only applicable for degree up to 5). Also, the new modular algorithm is applicable to fields defined as relative extensions; this is a very major improvement, since to this point there has never been any modular algorithm which handles relative extensions. (V2.21-4)

- *The Variety of an Ideal*

  Magma V2.21 contains major improvements to the algorithms for computing the affine variety of an ideal of $K[x_1, \ldots, x_n]$, where K is a finite field. Three new algorithms are now available for solving this problem:

  - *Wiedemann Algorithm:* Magma V2.21 contains a highly optimised implementation of the Wiedemann algorithm for computing the minimal polynomial of a large moderately sparse matrix (developed by Allan Steel). This is applicable to

computing the variety of a generic zero-dimensional ideal I of $R = K[x_1, \ldots, x_n]$, where K is a moderately sized finite field. The *degree* D of I is the dimension of the quotient algebra R/I, and as D grows large (particularly above 10000), computing the variety of I becomes very difficult since it involves computing a minimal polynomial of degree D. The Wiedemann algorithm is effective when it is impractical to store a fully dense $D \times D$ matrix in memory. Using this algorithm, one can solve a random instance over GF(65521) of the Cortois Minrank Challenge C where $(n, k, r) = (11, 9, 8)$, using the direct approach with the Wiedemann algorithm; here the degree $D = 259545$.

– *Keller-Gehrig Algorithm:* The second algorithm is similar to the Wiedemann algorithm in that it finds the minimal polynomial of a matrix to compute a variety of a zero-dimensional ideal, but it uses the Keller-Gehrig algorithm to compute this minimal polynomial by mapping the problem to several matrix multiplications. This method typically takes more memory than the Wiedemann algorithm but may be faster for dimensions in the thousands, particularly when a GPU is available.

– *Exhaustive Search Algorithm:* Magma V2.21 contains an *exhaustive search* algorithm, developed by Geoff Bailey, for solving quadratic multivariate polynomial systems over GF(2). For a system with n variables, the algorithm simply enumerates all 2^n possible assignments of the variables and determines the set of all such assignments which simultaneously satisfy all the polynomial equations.

  Despite the obvious exponential complexity of this brute force approach, the algorithm is highly optimised: for a quadratic input system, the algorithm covers 10^{10} possible solutions in 1.47 seconds on a typical 3.2GHz Intel Xeon processor and an arbitrary quadratic system with 40 variables is solved in 162 seconds.

  The algorithm is preferable to other algorithms for certain types of input system. For example, in the case of an arbitrary generic dense quadratic system with no structure (such as a random system), the exhaustive search algorithm is typically faster than both the Gröbner basis and SAT algorithms by a factor of more than 100. This is because all algorithms have exponential complexity for this type of input, but the associated constant is much better for the exhaustive search.

Group Theory

- *Finitely-Presented Groups*
  - The lastest version of Sebastian Jambor's `L2Quotients` package has been installed in Magma. This version applies to fp-groups having an arbitrary number of generators (the previous version applied only to 2-generator groups).
  - Sebastian Jambor's `L3Quotients` package is now available. Given a 2-generator finitely presented group, it computes all possible L3-quotients (or U3-quotients) over a finite field.
  - Derek Holt's package for working with finitely-generated subgroups of free groups (FS package) has much increased functionality. It now includes algorithms for calculating normalizers and centralizers of subgroups as well as testing conjugacy of subgroups.
- *Finite Groups*
  - *Recognition Algorithms for Classical Groups:* The ability to recognise a quasi-simple group and then to create an isomorphism between the given group and a standard version is important for various applications. In particular this capability is central to the construction of the composition tree for a group (see below). The term *standard generators* used below refers to the generators of the standard version.
    - New code, provided by E. O'Brien, constructs standard generators for matrix and permutation representations of classical groups defined over finite fields (constructive recognition).
    - Machinery for the black-box recognition for classical groups has been included which provides constructive recognition for a much larger range of groups. The package was provided by E. O'Brien.
    - Improved algorithms, based on work of P. Brooksbank, have been incorporated for recognising $SU(3,q)$, $SU(4,q)$ and $Sp(4,q)$. The code was provided by E. O'Brien.
  - *Composition Tree:* A major effort has been underway for almost 20 years to devise effective methods for computing with large matrix groups defined over finite fields. This effort, led by Charles Leedham-Green and Eamonn O'Brien and involving many group theorists, has now reached the point where it is powerful and reliable enough to solve problems of interest that cant be solved by other means. A central tool is an algorithm to compute a *composition tree* (CT) for a matrix or permutation group G. This datastructure is intended to be a basis for structural computation in groups for which it is not possible to find a base and strong generating set (BSGS). The version released in V2.21-4 is a major upgrade to previous versions. It succeeds in computing a CT for

a much wider range of groups and it is generally much faster than previous versions. It is possible to construct the composition tree for most groups in the Atlas of Finite Groups (possibly using a minimal degree matrix representation over a finite field). The new version has been successful in constructing the composition tree for groups having matrix representations of degree up to 5000.

– *Soluble Radical Methods:* Derek Holt has developed an algorithm which takes a composition tree for G and computes the characteristic series which is the basis of the soluble radical approach to the design of structure algorithms. Using this series a number of key structure algorithms have been implemented. In this release a number of intrinsics for computing key properties of a group appear for the first time and many improvements have been made to the older ones. At present functions implementing the soluble radical approach for a group with a composition tree are known as LMG (Large Matrix Group) functions. For brevity we shall refer to such a group as an LMG group. The upgraded composition tree package combined with the LMG tools makes it possible to compute structural information and character tables for groups that are beyond the limit for BSGS techniques. For example, it has been used to compute the character table of the group $2^{1+22}.Co_2$ in a degree 1025 representation over $GF(2)$. The order of the group is 354 883 595 661 213 696 000.

  * The character table algorithm of Bill Unger (based on Brauer's Theorem) has been generalised to work for LMG matrix groups.
  * The normal subgroups of an LMG group may be computed.
  * Given an LMG group, its subgroups having index less than some (modest) bound may be computed.
  * Various functions are provided to obtain information about the action of an LMG group G on the cosets of some subgroup H. These include constructing the image of the action and finding a transversal for H in G.

– *Maximal Subgroup Database:* Many algorithms proceed by reducing a problem for group G down to a similar problem for the non-abelian composition factors of G. For the calculation to succeed the problem must be soluble for any non-abelian composition factor of G. For example, such information is needed when computing the maximal subgroups or automorphism group of G. This release expands the list of the simple groups for which maximal subgroups are available. More precisely, they are now available for all simple groups having a permutation representation of degree not exceeding 10,000 excepting the alternating groups where the the maximal subgroups are available for all degrees up to degree 2499. The full list of groups may be found in the second part of this document.

- *Permutation Groups*

  - Structural algorithms for permutation groups relating to the socle have been extended to apply to groups having degrees up to 10^9 from the previous limit of 10^7. The functions affected include those for computing socle, chief series, simplicity tests, composition series, and composition factors.
  - The algorithm used to test whether a permutation group is either alternating or symmetric has been improved so that it now takes less time, particularly in the case of groups having high degree.
  - A backtrack algorithm has been developed to find a canonical element of a double coset, given any element of the double coset. This makes it possible to test membership and equality of double cosets in a permutation group.
  - The database of transitive groups has been extended by Derek Holt to include the transitive groups having degrees $33 - 47$ inclusive. Also included is code for identifying a transitive group, that is, determining its number in the database list.

Linear Algebra and Module Theory

- *Linear Algebra over Finite Fields*
  - A multi-threaded parallel version of matrix multiplication has been developed for matrices over GF(2). This often leads to a speedup in wall-clock time close to a factor of k. Multiplication of large matrices over GF(2^k) for $k > 1$ often maps to the GF(2) algorithm, so a similar speedup will be present in such cases.
  - The asymptotically-fast Keller-Gehrig algorithm for computing the minimal polynomial of a matrix over a field has been implemented. The algorithm uses $O(\log_2(N))$ matrix multiplications and one nullspace computation, where N is the dimension of the matrix, so for large N and for reasonably large base finite fields, the algorithm may be rather faster than the default algorithm, especially if a GPU is present.
- *Linear Algebra over $K[x]$*
  - An analogue for the LLL lattice basis reduction algorithm is now available for matrices with entries in $K[x]$, for K a field. The algorithm uses asymptotically-fast techniques when the degree of the polynomials in the input matrix is large and/or the dimension of the matrix is large.
  - A new asymptotically-fast algorithm for computing the Hermite Normal Form (HNF) of a dense matrix over $K[x]$ (K a field) has been developed, which builds on the above LLL algorithm for matrices over $K[x]$. This algorithm is dramatically faster than the previous algorithm for uniform dense matrices, in general. For example, computing the HNF with transformation matrix of a dense 100×100 matrix over GF(32003)$[x]$, where the input entries are random polynomials of degree 3, takes 0.6 seconds, which is 32 times faster than the previous algorithm.
  - New asymptotically-fast algorithms for computing the determinant, rank, nullspace, inverse, or Smith Normal Form of a matrix over $K[x]$ have been developed, based on the above HNF algorithm.
  - A new asymptotically-fast algorithm for computing the echelon form of a matrix over the rational function field $K(x)$ (K a field) has been developed, which again builds on the above LLL algorithm for matrices over $K[x]$. In general the speed-up achieved by this algorithm is even more dramatic than that of the previous algorithm. For example, computing the inverse of a dense 100×100 matrix over GF(32003)(t), where the input entries are random polynomials of degree 3, takes 3.7 seconds, which is 205 times faster than the previous algorithm.
  - New asymptotically-fast algorithms for computing the determinant, rank, nullspace, or inverse of a matrix over $K(t)$ have been developed, based on the above echelon form algorithm.

- *Linear Algebra over Real and Complex Fields*
  - Magma now includes methods based on Householder reflections (RQ decompositions) for various functions involving linear algebra over a real or complex field. The functionality includes kernel, rank, solution, RQ-decomposition, QL-decomposition, and pseudoinverse.
  - The standard functions for computing the determinant and inverse of square matrices now use this new code internally.
- *Lattices*
  - An analogue for the LLL lattice basis reduction algorithm is now available for matrices with entries in $K[x]$, for K a field.
  - Block Korkine-Zolotareff (BKZ) reduction is now available in Magma.
  - The internal interface to use the `LLL` implementation of Damien Stehle has been extended to include more cases (particularly number fields).

Representation Theory

- *KG-Modules*
  - The code for computing the irreducible and absolutely irreducible F_q-modules for a finite (non-soluble) group has been improved so that it is now more likely to be successful in finding larger degree modules. Further upgrades will appear in upcoming patch releases.
  - A modular algorithm has been developed for computing Hom-modules and endomorphism rings of KG-modules defined over algebraic number fields (V2.21-4).
- *Characters of Finite Groups*
  - A version of Bill Unger's algorithm for computing the table of irreducible complex characters has been implemented for large degree matrix groups. This uses the Composition Tree datastructure for a group together with Derek Holt's LMG code. At present there are some limitations but it is possible to compute character tables for matrix groups over $GF(q)$ having substantial degree.

System

- *NVIDIA GPU support*
  - The procedure `SetGPU` now sets whether Magma should use NVIDIA GPUs via CUDA when present. This is only relevant to a CUDA-enabled executable (which is downloaded as `magma.cuda.exe`) and is `true` by default in that case (so a GPU is used by default).

3 Documentation

New Handbook Chapters:

– Local Galois Representations (T. Dokchitser).

4 Algebraic Geometry

4.1 Schemes

New Features:

- There is a new intrinsic `WeightedAffinePatch` to construct and return affine patches for weighted projective space ambients and patch indices where the grading is greater than 1, along with the projective closure map back into the projective ambient. These patches are affine schemes but generally not plain affine ambients. They are not incorporated into the lower-level scheme machinery in the way that the existing standard affine patches are.
- The `IsSingular` and `IsNonsingular` tests, which could not previously be fully applied to schemes in projective ambients apart from ordinary and product projective space, have been extended to apply and give correct results over the entirety of a subscheme in weighted projective space. Here and above, weighted projective space means a projective space with a single grading consisting of positive integer weights.
- A `Complement` intrinsic has been added. This is similar to `Difference` but it completely removes all points of the subscheme to be excised rather than removing components to multiplicity one.

Changes and Removals:

- Schemes cannot now be constructed over rings which do not contain a one. (V2.20-6)
- Creation of invalid points on projective ambients over the quotient of a UFD, R, has been disallowed. When R is an integral domain or the relevant gcds are non-zero divisors, gcds are divided out of the coordinates if the gradings allow. After this preprocessing, the condition that each ideal generated by the coordinates of a redundant ideal (for the ambient) contains 1 is now checked. Problem reported by Benjamin Hutz.

Bug Fixes:

- A bug with lifting the origin with `IsolatedPointsLifter` (due to erroneous use of `Precision` rather than `AbsolutePrecision`) was fixed. Reported by A.-S. Elsenhans and J. Jahnel. (V2.20-4)

4.2 Analytical-hypersurface Singularities

New Features:

- A package to deal with corank 3 singularities and largely complete the Arnold classification code for isolated hypersurface singularities has been added. Just as for corank 2, there is an intrinsic `Corank3Case` that can be called directly or the user may access the new classification code through the top-level `NormalFormOfHypersurfaceSingularity` intrinsic. In the initial release, not all corank 3 families will be covered. The P, R and T families will be included along with some of the S families. The remaining S families along with the U and V families will be added in a secondary release next year.
- The $X_{k,r}$ family of corank 2 singularities has been added.

4.3 Algebraic Surfaces

New Features:

- A local blow-up desingularisation method for surfaces has been added to complement formal desingularisation. There is a new main intrinsic `ResolveSingularSurface`, replacing the old main intrinsic `ResolveProjectiveSurface` (see also changes below), which allows a choice between desingularisation method, and desingularisation data (of old or new type) is now cached with the surface. The blow-up method currently only works for surfaces with only point singularities, but the surface may be in any ambient of any characteristic.
- For blow-up desingularisation, there are a number of new intrinsics to compute intersections and multiplicities of the blow-up divisors above singular points in pullbacks or strict transforms of divisors on the surface or in pullbacks of differential forms on the surface. `DifferentialMultiplicities`, `Multiplicities`, `MultiplicitiesAndIntersections` and `CanonicalIntersection` are the main ones. Also, `IntersectionMatrix` gives the matrix of intersection numbers for the irreducible blow-up divisors over a singular point.
- Intrinsic `LinearSystemDivisorRestriction` allows the computation of the subsystem of a linear system consisting of the sections that vanish to specified multiplicity on the blow-up divisors.
- Most of the intrinsics using a formal surface desingularisation have been extended to also use the new blow-up desingularisations (see changes below).

Changes:

- The old `ResolveProjectiveSurface` has been renamed `FormallyResolveProjectiveHypersurface` (but is now largely redundant, anyway).
- The parameters `FormalDesing` to pass a formal desingularisation data list to various surface intrinsics (e.g. `HomAdjoints`) have been removed since such data is now cached internally with the surface when computed. They have been replaced with Boolean `UseFormalDesing` parameters to give the choice between the two desingularisation methods now available.
- The intrinsics `Geometric/ArithmeticGenusOfDesingularization`, `ParametrizeRationalSurface`, `PlurigenusOfDesingularization`, `ClassifyRationalSurface`, `HomAdjoints` and `IsRational` have been extended so that they can now use either formal or blow-up desingularisation data.
- `HasOnlySimpleSingularities` has been partially rewritten to make proper use of Milnor numbers in computing the precise singularity type. This avoids unnecessary power series expansion once the possible family type (A,D or E) has been determined and will greatly speed up execution in many cases.

5 Arithmetic Geometry

5.1 Elliptic Curves

5.1.1 Elliptic Curves over the Rational Field

New Features:

– A function `Saturation(E)` has been added, in addition to the existing function `Saturation(E,N)` which saturates at the specified range of primes.

Changes and Removals:

– The functions `RankBounds`, `Rank`, `MordellWeilGroup`, `Generators` and other functions related to the group of rational points on an elliptic curve over **Q** or a number field have undergone a major internal change. These functions now make use of the full power of machinery in Magma. For curves over **Q** this typically means (as necessary) 2-descent, Cassels-Tate pairings, 4-descent, 8-descent, 3-descent, as well as other techniques, such as analytic rank and Heegner points when the conductor is not too large.

The functions now additionally return one or two booleans following the previous return values (which are unchanged). `MordellWeilGroup` returns two booleans: the first is `true` iff the rank of the group returned is known to be the rank of the curve. The second is `true` iff the group returned is known to be the full Mordell-Weil group.

The warning message which was sometimes printed in unresolved cases is no longer printed.

The optional parameter `HeightBound` retains the same meaning (although its effect will be slightly different now that local saturation techniques are also used in the algorithm). A parameter `Effort` has been added, which currently has the following effect. For curves over **Q**: when the effort is 1, the algorithm terminates before attempting the hardest methods (`FourDescent` is used but not `EightDescent` or `ThreeDescent`), otherwise all methods are used. For curves over number fields: all available methods are always used, and the `Effort` is a linear multiplier that controls the time spent in searching for points in some of the methods.

– The function `MordellWeilShaInformation` is a slightly lower level interface to `MordellWeilGroup` which also prints information during runtime and returns information about the Tate-Shafarevich group. There are minor changes here. A parameter `Effort` has been added with the same effect as in the preceding item. The function now attempts to saturate fully, which means the user must set the `HeightBound` parameter in cases where this is infeasible due to the bound on the difference between naive and canonical heights being too large. (In future releases it is hoped to use more advanced algorithms to overcome this problem.)

– An algorithm for computing a rank bound for a curve over **Q** with a 2-isogeny, by performing several higher descent steps, has been contributed by Tom Fisher. A summary table is printed after each step, if the verbose flag `cbrank` is set to 1, which displays the rank bounds obtained so far. The function is `TwoPowerIsogenyDescentRankBound`. By default it does 5 descent steps; for curves with full torsion it can do 6 steps, which is equivalent to performing full 8-descent on the curve. The routine can also be used to obtain the 4-coverings returned by standard `FourDescent`.

– Machinery for 5-descent has been contributed by Tom Fisher. The interface is almost identical to 3-descent, so these functions have not been documented individually. The main functions are `FiveTorsionPoints`, `FiveTorsionMatrices`, `FiveSelmerElement`, `FiveDescentCoveringCurve`.

– A function `FourDescent(E)` has been added for convenience, which automates the process of doing TwoDescent and then FourDescent on the 2-coverings, etc.

– The Cremona database of elliptic curves of small conductor now extends to conductor 360,000.

Bug Fixes:

– A bug (runtime error) in `FourDescent` has been fixed. The bug occurred in rare cases when the option `RemoveTorsion` was used. (V2.20-10)

– A problem with odd symmetric powers of L-functions of elliptic curves with complex multiplication was fixed (the product did not contain the L-function of the elliptic curve itself). (V2.20-10)

5.1.2 Elliptic Curves over Number Fields

Changes and Removals:

– The interface to all high level functions related to rational points (Mordell-Weil groups) is now the same for curves over number fields as for curves over $\mathbf{Q}$. These functions are described in the preceding section.

Bug Fixes:

– A bug in `HasComplexMultiplication` that caused runtime errors in some cases has been fixed. Reported by W. Moore. (V2.20-7)

– `EllipticCurveSearch` now works for $\mathbf{Q}(i)$ and $\mathbf{Q}(\zeta_3)$. Reported by John Cremona. (V2.20-10)

– A runtime error has been fixed in reduction of genus one models over certain number fields. This affected MordellWeilShaInformation for those fields. Reported by Kazuo Matsuno. (V2.20-7)

5.2 Hyperelliptic Curves and Jacobians

New Features:

– The computation of semistable models at bad odd primes for arbitrary genus hyperelliptic curves is now available and improves the L-series machinery.

– Computing conductors and Euler factors for hyperelliptic curves over the rationals, number fields, and p-adic fields is also implemented.

– Finally, L-series of hyperelliptic curves over number fields are available.

Bug Fixes:

– Several bugs (runtime errors and infinite loops) have been fixed in the package for heights on Jacobians. Reported by M. Stoll. (V2.20-2)

– The code to compute p-minimal models was not following the same logic as the `IspMinimal` intrinsic, causing intrinsics such as `pMinimalWeierstrassModel` to return models which were in fact not p-minimal. Reported by A. Sutherland. (V2.20-8)

5.3 Hypergeometric Motives

New Features:

- The `HodgeStructure` of odd weight data is now directly obtainable, without specifying a t-value. (V2.20-3)
- The initialisation of large degree data is now more efficient, as large cyclotomics are avoided and the GammaArray length is not artificially lengthened during the pre-computation. (V2.20-3)
- A vararg `Weight01` has been added to the `LSeries` creation intrinsic, which takes the Tate twist (via `Translate`) of the LSeries to make it have (motivic) weight 0 or 1. (V2.20-3)
- A vararg `QuadraticTwist` has been added to the `LSeries` creation intrinsic, which can be specified either as a nonzero rational or as a real Dirichlet character, or alternatively via the boolean true, when Magma will then use a default twisting factor that aims to help some deformation theory work out. (V2.20-3)
- Some functionality for non-disjoint hypergeometric data is now available. In particular, the `LSeries` intrinsic now returns two L-series, the first corresponding to the disjoint data, and the second corresponding to an Artin representation for the common data. (V2.20-3)
- Magma now contains a database (of wild prime information) for $t = 1$ degenerations. This contains all examples up through degree 6. (V2.20-4) and (V2.20-7) and (V2.20-8)
- The LSeries intrinsic for hypergeometric data now has a `ZetaOrder` vararg that allows the user to specify that the L-series is expected to have a factor of the Riemann ζ-function (possibly shifted) raised to the given power.(V2.20-4)
- The `Character` of hypergeometric data at a given t-value (including $t = 1$) can now be obtained directly. (V2.20-7)
- There is now a WARNING when Magma ignores specified `BadPrimes` info when it already knows an object associated to the hypergeometric data. Similarly with the $t = 1$ database mentioned above. (V2.20-8)
- The L-series of a hypergeometric datum now has a `SaveEuler` option that allows the user to specify that Euler factors up to a given bound should be saved. This is useful when the L-functions themselves are then used in other constructions (tensor product, or symmetric power), as the coefficients of the underlying object need only be computed once. (V2.20-8)
- A fix has been applied for `BadPrimeData` with wild primes.

Changes:

- The default printing has been changed to use cyclotomic indices. The alpha/beta printing (which is not so useful over the rationals) can be obtained via a vararg Print when initialising the data. (V2.20-3)
- The automatic swapping of ALPHA and BETA was removed (also affecting `Twist`). (V2.20-4)
- The `Weight01` vararg to `LSeries` has been modified to allow a weight dependent on the number of trivial cyclotomics in the data. (2-20.7)

Bug Fixes:

- A runtime error is now triggered for tame Euler factors that are too hard to compute to high precision (rather than a crash). (V2.20-3)

– Some minor problems with deformations at $t = 1$ and the `Identify` function were patched. Similarly with `EulerFactor` at $t = 1$. (V2.20-4)

– The `EulerFactor` intrinsic was returning polynomials over the rational field instead of the integers in some cases (particularly at multiplicative primes). (V2.20-7)

– A bug was fixed, concerning computing the central Hodge sign(s) when the ALPHA and BETA arrays were switched from the original normalization that was used in previous Magma versions. (V2.20-8)

5.4 L-Series

New Features:

– The case of a tensor product when a bad prime has both constituents with a degree 1 factor is now handled automatically (thanks to Tim and Vladimir Dokchitser). (V2.20-3)

– The `RootNumber` at infinity of a Hodge structure is now available, returned as an element of the 4th cyclotomic field. (V2.20-3)

– Calling `LSeries` on the rationals will now give RiemannZeta. (V2.20-3)

– Division by the trivial L-series is now possible in any weight. (V2.20-7)

– Utility intrinsics to `ChangeLocalInformation` and `CopyCoefficients` between L-functions are now provided, to ease the difficulties in numerically determining bad Euler factors. (V2.20-7)

– The `HodgeStructure` of various objects have been added, such as elliptic curves, Hilbert modular forms, and Artin representations. (V2.20-7)

– An intrinsic for the `Degree` of a `HodgeStructure` has been added. (V2.20-7)

– The L-series of a 1-dimensional modular symbol space is now possible. (V2.20-4)

– A new intrinsic `BadPrimeData` has been added that gives the information for primes that divide the conductor. (V2.20-8)

– Multiplication by the trivial L-series is now possible in any weight. (V2.20-8)

– Multiplication of `RiemannZeta` with itself should now work. Similarly with other products involving L-functions with simple poles. (V2.20-8)

– A new intrinsic `Symmetrization` has been introduced for L-series. This is still not fully implemented, but should allow significantly more examples to be computed than previously. Furthermore, orthogonal and symplectic versions are available where applicable. (V2.20-8)

– Twists of elliptic curves by Artin representations now have no restrictions on ramification.

– The `MotivicLSeries` package now allows faster computation of `CheckFunctionalEquation` in some circumstances. (V2.21-4)

Changes:

– The HodgeStructure printing has changed, so that repeated (p,q) pairs appear with a multiplicand in front. (V2.20-3)

– An experimental intrinsic `CFENew` has been added that uses a dynamic method (relative precision of summands versus sum) to determine how many terms to use in checking the functional equation. Comparatively the intrinsic `CheckFunctionalEquation` simply uses `LCfRequired` to get the required number of coefficients (usually a good factor larger). (V2.20-7)

– The convention for `TateTwist` has been reversed in direction. (V2.20-8)

Bug Fixes:

– The TensorProduct of two LSeries with integral coefficients should now keep this integrality at the EulerFactor for every prime. There was a bug in the case where one constituent had degree 1. Reported by D. Roberts. (V2.20-3)

– A printing error with HodgeStructure has been fixed. (V2.20-4)

– A bug with EulerFactor of an L-series of a modular form has been corrected. (V2.20-4)

– A fix to `TensorProduct` has been made, to ameliorate problems when the user has specified a trivial Euler factor (1) of one of the constituents as an integer rather than pedantically as a polynomial. (V2.20-4)

– A bug was fixed in the general `SymmetricPower` intrinsic, which had caused a crash due to a incorrect argument passed to `HodgeStructure`. This intrinsic has also had its `BadEulerFactors` vararg be replaced by the more common BadPrimes, though the former is still retained for compatibility. (V2.20-4)

– The `TensorProduct` intrinsic with two arguments can now take the `BadPrimes` info as a vararg, rather than having this be a third argument (as with `ExcFactors` the latter is also still available). (V2.20-4)

– A bug with `TensorProduct` concerning Artin representations has been fixed. This could also afflict elliptic curves over number fields which internally use these. Noted in part by J. Cremona and W. Moore. (V2.20-7)

– Various issues with precision have been remedied. (V2.20-7)

– `Translate` now tries to translate the residues (in addition to poles). (V2.20-7)

– Some more minor changes to internal precision limits have been made for L-functions of larger degree and weight. (V2.20-8)

– An error with SymmetricPower of a Hodge structure was corrected. (V2.20-8)

– An issue with passing precision to a tensor product of imprimitive L-functions has been resolved. (V2.20-8)

– An issue with EulerFactor of non-rational modular forms was fixed. (V2.20-8)

– The base ring of EulerFactor should now be the integers when the data allow it (previously, it often was defined over the rationals). (V2.20-8)

– The O-term in `LTaylor` is now correct when the `Valuation` is nonzero. (V2.20-8)

5.5 Admissible Representations

Changes:

– The intrinsics `GaloisRepresentation` and `WeilRepresentation` now return local Galois representations, rather than four pieces of data.

6 Arithmetic Fields (Global)

6.1 Dirichlet and Hecke Characters

Changes:

- Real Dirichlet characters over **Q** now print as "Kronecker character", which should be more clear than the previous. (V2.20-8)

Bug Fixes:

- A bug with 'meet' of character subgroups was fixed. (V2.20-8)

6.2 Algebraic Number Fields

New Features:

- There is a new implementation of size reduction of an element by multiplication by units of the maximal order. This size reduction is now done in all situations where one would expect it to be done, for example in choosing a generator for a principal ideal and so on.
- A facility for passing between Abelian fields (`FldAb`) and Dirichlet/Hecke characters has been implemented, at the request of M. Kirschmer and aid from C. Fieker.
- For a "non-simple" field or order, i.e. defined by several polynomials, the simple fields/orders corresponding to each polynomial are returned by `Components`.
- A prime integer can now be directly decomposed in an order in a relative number field.
- An infinite place of a number field can now be decomposed in a fields of type `FldOrd` as well as `FldAlg`.
- Extended types can now be used with places and divisors of number fields and ideals of orders of number fields.

Changes and Removals:

- The `Magma` level printing for orders has been fixed. Orders defined as transformations of other orders are printed in a way that allows reconstruction.
- For a field K defined by a non-monic or non-integral polynomial, `K.1` is now the root of the defining polynomial (previously it was zero).

Bug Fixes:

- Various problems with computations involving units have been fixed. In occasional instances, a crash (internal error) was encountered. In other occasional instances, an incorrect answer was returned, such as a false unit, or an incorrect ideal generator.

  Algorithms for unit groups that were selected by parameters "Dirichlet", "Mixed", "Relation", "Short" are now never used (the parameters are ignored). Use of these algorithms sometimes resulted in incorrect answers. (V2.20-5 and V2.20-6)

– A bug has been fixed concerning precision of the real and complex images of elements (or bases of orders). The complexity of the change of basis transformation by which the element is defined is now taken into account. The bug made certain computations impossible, including the class group of some fields of moderate or large degree.

  Corrections have been made to the real precision used in computing the LLL (basis) of an order. Insufficient precision lead to a result which was far from LLL in some (unusual) cases, and this caused crashes in computing class groups. (V2.20-5 and V2.20-6)

– A bug has been fixed in ClassGroup affecting the WINDOWS version (causing a crash or very bad performance). Reported by K. Matsuno. (V2.20-8)

– Some anomalous behaviour in the hashing of number field elements and order elements has been fixed. This manifested in poor performance in some algorithms. (V2.20-6)

– A bug in the construction of an order from a sequence of polynomials has been fixed.

– A bug in the coercion of elements between number fields has been fixed.

– A bug with `Distance` and `Diameter` has been fixed (the maximum of said intrinsics was always 1). The old behaviour can be obtained via a new `Max` vararg (setting it equal to 1). Reported by C. Neurohr.

6.2.1 Cyclotomic Fields

Bug Fixes:

– Some problems with linear algebra over the first `CyclotomicField` have been fixed. (V2.20-7)

6.3 Characters and Artin Representations

Changes:

– For Artin representations, there are now faster implementation for p-adic computations at bad primes.

– The printing for Artin representations now print in a way that includes the group through they factor.

– Real Dirichlet characters over $\mathbf{Q}$ now print as "Kronecker character", which should be more clear than the previous. (V2.20-8)

Bug Fixes:

– A bug with degree 0 representations over the rationals was fixed. (V2.20-3)

– `IsZero` can now be applied to an Artin representation directly. (V2.20-3)

– The sum of an empty sequence of Artin representations over $\mathbf{Q}$ now works. (2-20.3)

6.4 Algebraic Function Fields

New Features:

- A `ChineseRemainderTheorem` intrinsic has been added. (V2.20-4)
- Extended types can now be used with ideals of orders of function fields to specify whether the ideal is an order of a function field which is represented as an extension of the rational function field or another algebraic function field.

Changes and Removals:

- The defining polynomials of equation orders of function fields whose defining polynomials are non-monic or non-integral have been improved. (V2.20-4)
- A parameter `Strict` has been provided to `StrongApproximation`.
- A `ChineseRemainderTheorem` intrinsic has been added taking a sequence of places, a sequence of elements and a sequence of integers.

6.5 Abelian Extensions of Function Fields

New Features:

- The computation of `SMaximalOrders` of degree p^n extensions defined by a Witt vector has been added. (V2.20-2)
- Both finite and infinite maximal orders of the cyclic extension defined by a Witt vector can be computed using `MaximalOrders` with the Witt vector as input.

Changes:

- The computation of maximal orders of Artin–Schreier—Witt extensions has been sped up by using the chinese remainder theorem instead of strong approximation when this is faster. (V2.20-4)
- Arithmetic with Witt vectors has been made more efficient. (V2.20-4)
- The computation of maximal orders of abelian extensions and in particular cyclic extensions of function fields has been improved.

6.6 Galois Groups

New Features:

- The `GaloisGroup` of a non-simple extension of $\mathbf{Q}$ can now be computed directly and without computing the isomorphic extension by one polynomial.

Changes and Removals:

- The Galois group algorithm treats subfields now non-recursively. This makes the code faster and more stable.

– In case of a primitive Galois group the algorithm will factor the 2-set resolvent to get a better starting point.

– In case the Galois group has a unique minimal block system of block size at least 3 the algorithm will compute the Galois group of an additional auxiliary field first, to get a better starting point. This results in a faster treatment of such cases.

– The internal generation of invariants has been optimized. The algorithm will handle any irreducible polynomial up to degree 47 and many of even larger degree.

– The verbose printing has been brushed up.

Bug Fixes:

– A bug in the computation of the `GaloisGroup` of a reducible polynomial has been fixed.

– A bug involving incorrect handling of Tschirnhausen transformations in the Galois group algorithm has been fixed.

7 Arithmetic Fields (Local)

7.1 p-adic Rings and their Extensions

New Features:

- Galois groups and splitting fields are now expedited via new computational methods by T. Dokchitser.
- One can now compute the `PowerRelation` for an element of $\mathbf{Q}_p$. (V2.20-3)

Bug Fixes:

- `IsZero` has been fixed for p-adics. `IsZero(Zero(K))` is now true. (V2.20-4)
- Checking equality with zero for p-adics, in expressions such as `0 eq pAdicRing(11)!0` now are `true`, even when assigning the result to a variable. Previously, in the latter case (of assigning a result), the Magma interpreter would try to use special is_zero code that would incorrectly return `false`. (V2.20-4)
- A number of problems with `AllExtensions` and `NumberOfExtensions` for a p-adic field (or ring) have been resolved. In particular, the "D" vararg has been changed to now be the valuation of the discriminant. Noted in part by T. Dokchitser.

7.2 General Local Fields

New Features:

- The intrinsics `Norm`, `Trace` and `MinimalPolynomial` are now provided for elements of general local fields. These can also be calculated with respect to a given coefficient field.

7.3 Newton Polygons

Changes and Removals:

- A fix has been made to Newton Polygons constructed from a polynomial over a local ring with zero coefficients. Such zero coefficients no longer correspond to a finite point on the polygon. (V2.20-10)

8 Basic Rings and Fields

8.1 Integer Ring

Bug Fixes:

- The integer factorization had a couple of fixes applied, to ensure that the B1 limit for ECM was small enough (else all the prime factors could often be found at the same time, not leading to a nontrivial splitting). (V2.20-9)

8.2 Real and Complex Fields

New Features:

- The print level `Magma` is now handled properly when printing real numbers and complex numbers.

Changes and Removals:

- The deprecated intrinsics `LinearRelation` and `PowerRelation` of real or complex numbers now only use the "LLL" option, and ignore the "Hastad" option in all cases. The latter was quite buggy and never achieved as good as results in any event. The newer intrinsics to use (present for many versions by now) are `IntegerRelation` and `MinimalPolynomial` (which are now documented in the handbook). (V2.20-5)
- The `MinimalPolynomial` intrinsic has been changed to always return a polynomial with positive leading coefficient. (V2.20-5)

Bug Fixes:

- Two fixes to `Roots` of a complex polynomial were made, one to delimit an infinite loop, and the other to try to handle multiple roots more gracefully. (V2.20-3)
- A crash (memory corruption) with polynomials of degree 1 was also fixed. (V2.20-5)
- An internal issue with the norm of complex number with differing real and imaginary relative precisions has been patched. However, for some intrinsics such as the `Polylog`, the user must still some exercise caution in the arguments they pass to the function. (V2.20-5)
- The intrinsic `HypergeometricSeries2F1` has a number of problems, noted by J. Detrey. Firstly the answer is wrong in cases where the parameters induce a zero in the Gamma-quotient (pole in the denominator) from the transformation law. Secondly, a bug (division by zero) can occur when the parameters form nonpositive integers upon transformation. However, even upon fixing this directly, there would still be numerical instability when the transformation leads to parameters which are nearly nonpositive integers. The first problem has been patched, while the second one has not yet been addressed in full. (V2.20-9)
- A crash with `Psi` (equivalently `LogGamma`) at arguments with negative real part has been fixed. Reported by E. Crane. (V2.20-10)

8.3 Polynomial Rings

New Features:

- The algorithm to factorise a polynomial $f \in K[x]$ where K is a number field has been significantly improved, leading to general speedups. In particular, there is often a large speedup when f has small degree and K has large degree, or f has large degree and K has small degree.
- The algorithm to compute the roots of a polynomial $f \in K[x]$ where K is a number field has been been sped up (beyond the improvements to general factorisation above). The parameter `Max` for the `Roots` function is now also handled efficiently when factoring over number fields.
- The modular algorithm for univariate polynomial GCD in $K[x]$ where K is a number field, has greatly improved (particularly for the case where the leading coefficients of the input polynomials are large).
- The modular resultant algorithm for multivariate polynomials over $\mathbf{Z}$ or $\mathbf{Q}$ has been significantly improved.
- The multivariate resultant algorithm has been greatly sped up by use of Bezout matrices (instead of the Sylvester matrices) when there is a larger number of terms but the degrees of the input polynomials in the main variable are quite small.
- The ChangeRing intrinsic can now be applied to a multivariate polynomial. Requested by A. Sutherland. (V2.20-7)

Bug Fixes:

- A hang in multivariate factorisation over the integers has been fixed. Reported by T. Fisher. (V2.20-9)
- A bug in `MonomialCoefficient` for polynomial quotient rings has been fixed. Reported by E. Rains. (V2.20-8)

9 Commutative Algebra

9.1 Ideal Theory and Gröbner Bases

Magma V2.21 contains major improvements to the algorithm to compute the affine variety of an ideal of $K[x_1, \ldots, x_n]$, where K is a finite field. This consists of 3 new algorithms, as follows:

1. *The Wiedemann Algorithm*

   Magma V2.21 contains a highly optimised implementation of the Wiedemann algorithm to compute the minimal polynomial of a large moderately sparse matrix (developed by Allan Steel). This is applicable to computing the variety of a generic zero-dimensional ideal I of $R = K[x_1, \ldots, x_n]$ where K is a moderately sized finite field. The *degree* D of I is the dimension of the quotient algebra R/I, and as D grows large (particularly above 10000), computing the variety of I becomes very difficult because it involves computing a minimal polynomial of degree D. The Wiedemann algorithm is effective when it is impractical to store a fully dense $D \times D$ matrix in memory.

   Using this algorithm, one can solve a random instance over GF(65521) of the Cortois Minrank Challenge C where $(n, k, r) = (11, 9, 8)$, using the direct approach with the Wiedemann algorithm; here the degree $D = 259545$. An instance of this challenge had apparently never been explicitly solved before.[1] A current run with Magma V2.21, which uses just one Intel E5-2687W (3.1GHz) core and a Tesla C2075 GPU, involves two stages:

   - *Dense* F_4: 7.4 hours [computes the grevlex Gröbner basis of the zero-dimensional ideal; the GB has 51275 polys with up to 48620 terms each].
   - *Wiedemann*: 23.2 hours [computes the minimal polynomial of a 259545 by 259545 matrix with density 13.4%].

   See:

   `http://magma.maths.usyd.edu.au/users/allan/densef4/#CC`

   for more details.

2. *The Keller-Gehrig Algorithm*

   This algorithm is similar to the Wiedemann algorithm in that it computes the minimal polynomial of a matrix to compute a variety of a zero-dimensional ideal, but it uses the Keller-Gehrig algorithm to compute this minimal polynomial by mapping the problem to several matrix multiplications (see the Matrices section). This method typically takes more memory than the Wiedemann algorithm but may be faster for dimensions in the thousands, particularly when a GPU is available.

[1] See `http://www-polsys.lip6.fr/~jcf/Papers/FSS10.pdf` for more discussion.

3. *The Exhaustive Search Algorithm over* GF(2)

   Magma V2.21 contains a special *exhaustive search* algorithm for solving quadratic multivariate polynomial systems over GF(2) by exhaustive search (developed by Geoff Bailey). For a system with n variables, the algorithm simply enumerates all 2^n possible assignments of the variables and determines the set of all such assignments which satisfy all the polynomial equations in parallel.

   Despite the obvious exponential complexity of this brute force approach, the algorithm is highly optimised: for a quadratic input system, the algorithm covers 10^{10} possible solutions in 1.47 seconds on a typical 3.2GHz Intel Xeon processor and an arbitrary quadratic system with 40 variables is solved in 162 seconds.

   The algorithm is preferable to other algorithms for certain types of input system. For example, for an arbitrary generic dense quadratic system with no structure (such as a random system), the exhaustive search algorithm is typically faster than both the Gröbner basis and SAT algorithms by a factor of more than 100. This is because all algorithms have exponential complexity for this type of input, but the associated constant is much better for the exhaustive search.

New Features:

- Setting the parameter `Al := "Wiedemann"` for the function `Variety` selects the Wiedemann algorithm, which is useful for large to very large degree ideals (typically 1000 to a few hundred thousand) over finite fields of moderate size.
- Setting the parameter `Al := "KellerGehrig"` for the function `Variety` selects the Keller-Gehrig algorithm, which is useful for large degree ideals (typically over 1000) over any type of finite field.
- Setting the parameter `Al := "ExhaustiveSearch"` for the function `Variety` selects the exhaustive search algorithm, which is applicable for ideals over GF(2) with 12 to 61 variables.
- The FGLM basis change algorithm for generic zero-dimensional ideals has been greatly sped up. The Keller-Gehrig algorithm can now also be used within this algorithm, and is selected by setting the parameter `Al := KellerGehrig"` for the procedure `Groebner` and related functions.
- The dense F_4 algorithm has been greatly sped up in general for several classes types of ideals over finite fields where there is a large amount of reduction to zero. For the Minrank Challenge C (see above), the `grevlex` GB is computed about 2.3 times faster than the dense F_4 algorithm in V2.20.
- A major new algorithm for interreduction of a set/sequence of polynomials has been developed, based on the Faugere F_4 algorithm. The algorithm can be selected by setting the new parameter `Faugere` to `true` in the function `Reduce`. This algorithm is dramatically faster than the default algorithm for many inputs.
- The general F_4 algorithm has been sped up on Intel/AMD processors with AVX support (using special AVX instructions) for ideals defined over the fields GF(2^k) for $k > 1$.
- The dense variant of the F_4 algorithm now allows a larger number of variables and higher degree than previously: the limit on variables is no longer 128 and the limit on the degree is no longer 4. (V2.20-2)
- The primary decomposition algorithm has been improved for ideals with gradings (by better exploitation of a grading when present).

– A major speedup has been introduced into the linear algebra phase of the F_4 algorithm for computing Gröbner bases. For example, over a small-prime finite field, the speedup for computing the Cyclic-10 GB is about a factor of 4 and the speedup for computing the Katsura-12 GB is about a factor of 20. Currently the improvement only applies to ideals over small-prime finite fields, but it is hoped that it will be extended to other kinds of fields soon. (V2.21-4)

– A major new asymptotically-fast modular algorithm has been developed for computing Gröbner bases of ideals defined over algebraic number fields. The new modular algorithm handles fields of arbitrary degree (while the previous modular algorithm was only applicable for degree up to 5). Also, the new modular algorithm is applicable to fields defined as relative extensions; this is a very major improvement, since to this point there has never been any modular algorithm which handles relative extensions. (V2.21-4)

Bug Fixes:

– A hang in `EliminationIdeal` has been fixed. Reported by M. Heusener.

– A bug in `IsPrime` for certain classes of ideal has been fixed. Reported by A. Laface. (V2.20-5)

– A crash involving ideals with a large number of variables has been fixed. Reported by M. Grassl. (V2.20-6)

– A crash in the F_4 algorithm with huge matrices has been fixed. Reported by M. Grassl. (V2.20-8)

– A hang in Gröbner basis computation over number fields has been fixed. Reported by T. Fisher. (V2.20-9)

9.2 Modules over Multivariate Polynomial Rings

New Features:

– The function `Reduce` is now fully supported for a set/sequence of elements of a module over a multivariate polynomial ring. The parameter `Faugere` is also now provided for a F_4-type reduction algorithm (see Subsec.9.1 above).

– Coercion has been improved for modules (so more cases are handled).

Bug Fixes:

– Separate crashes in `Kernel` for modules over polynomial rings have been fixed. Reported by G. Brown and L. Carafi.

– A crash in `FreeResolution` for modules over non-prime fields of characteristic 2 has been fixed. Reported by I. Soprunov. (V2.20-7)

9.3 Affine Algebras

New Features:

– The construction of the representation matrix of an element of an affine algebra has been sped up in general.

10 Groups

10.1 Finite Groups

Additions:

- The collection of simple groups where the maximal subgroups and automorphism groups can be computed has expanded. The current list is as follows. All simple groups of order less than 1.6×10^7, A_n for $n \leq 2499$, $L_2(q)$, $L_3(q)$, $L_4(q)$, $L_5(q)$, $L_6(q)$ and $L_7(q)$ for all q, $U_3(q)$ for all q, $U_4(q)$ for all q, $S_4(q)$ for all q, $L_d(2)$ for $d \leq 14$, and the following groups: $U_5(3)$, $U_6(2)$, $U_7(2)$, $U_8(2)$, $S_6(3)$, $S_6(4)$, $S_6(5)$, $S_8(2)$, $S_8(3)$, $S_{10}(2)$, $S_{12}(2)$, $O_8^+(2)$, $O_8^+(3)$, $O_8^+(4)$, $O_{10}^+(2)$, $O_{10}^+(3)$, $O_{12}^+(2)$, $O_7(3)$, $O_7(5)$, $O_9(3)$, $G_2(4)$, $G_2(5)$, ${}^3D_4(2)$, $Sz(32)$, ${}^2F_4(2)'$, McL, J_3, HS, M_{24}, Co_2, Co_3, He, Fi_{22}, Ru, Suz, ON.
- Utilities for generating group names, and constructing groups from names, are now available.
- Intrinsics for recognition of quasi-elementary and **Q**-groups, and virtual permutation characters, have been added.

Changes:

- The `CayleyGraph` and `SchreierGraph` functions now have parameters `Labelled` and `Directed` to control whether the graph returned is labelled and directed.

10.2 Classical Groups

New Features:

- A `ClassicalConstructiveRecognition` function is included, that constructs standard generators for matrix and permutation representation of classical groups defined over finite fields.
- New function `RecogniseSp4` implements Brooksbank's Sp(4,q) recognition algorithm. This function should be used in preference to `RecogniseSp4Even`.
- Improved versions of `RecogniseSU3`, `RecogniseSU4`, based on original implementations of Brooksbank, have been installed.

10.3 Finitely Presented Groups

New Features:

- The latest version of Sebastian Jambor's `L2Quotients` package has been installed in Magma. This version applies to fp-groups having an arbitrary number of generators (the previous version applied only to 2-generator groups).
- Sebastian Jambor's `L3Quotients` package is now available. This computes all possible L3-U3-quotients (over a finite field) of a 2-generator finitely presented group. The main function is `L3Quotients`. Extra functionality is provided by `GetMatrices`, `SpecifyCharacteristic`, `IsFinite`, `AddGroupRelations`, and `AddRingRelations`.

– Derek Holt's package for computing with finitely generated subgroups of a free group has increased functionality. Functions in this package include `FSIndex`, `FSFiniteIndex`, `FSIsIn`, `FSIsSubgroup`, `FSEqual`, `FSFreeGenerators`, `FSMeet`, `FSNormaliser`, `FSIsConjugate`, and `FSCentralizer`.

Changes:

– In the package for computing with finitely generated subgroups of a free group, the following functions have been withdrawn and replaced. `FreeGroupIndex`, `FreeGroupIsIn`, `CosetTableToDFA`, `SubgroupsMeet`, `FreeGenerators`.

– The `GeneratorOrder` parameter of the `AutomaticGroup` function now accepts a sequence of integers (generator numbers and their negatives) as input, as well as accepting a sequence of generators and their inverses. (V2.20-7)

– The data stored in a `GrpFPHomsProc`, as constructed by `HomomorphismsProcess`, no longer includes a list of all homomorphisms found during the course of the process, as this was causing excessive memory use. As such, the two functions `Complete` and `Homomorphisms` can no longer be applied to these objects. To get a complete list of homomorphisms found by the process the user must now do this manually, repeatedly using `Homomorphism` to extract the current homomorphism from the process, then storing it, using `NextElement` to advance the process, until `IsEmpty` returns true.

Bug Fixes:

– A bug in `PCGroup` applied to an fp-group has been fixed. The bug caused incorrect results to be returned when `PCGroup` was applied to an fp-group which did not have a presentation computed. (V2.20-4)

– A crash in computing the order of an fp-group has been fixed. Bug reported by Steve Humphries. This has increased the stability of fp-group calculations generally. (V2.20-6)

– A problem with `IsSatisfied` when the 2nd argument is empty has been fixed. If the length of the 2nd argument does not equal the number of generators of the fp-group relevant to the first argument, then there will now be a runtime error in all cases, not just some. (V2.20-9)

– A crash in `LowIndexSubgroups` has been fixed. This was related to giving a maximum subgroup index of zero. (V2.20-10)

10.4 Polycyclic Groups

Bug Fixes:

– Some problems computing the centre have been fixed.

– A crash when testing an inconsistent presentation for consistency has been fixed.

10.5 Matrix Groups Over Finite Fields

Derek Holt's Large Matrix Group (LMG) package, which uses the Composition Tree (CT) datastructure as constructed by Eamonn O'Brien and collaborators, has increased functionality. The CT datastructure is used as a replacement for the BSGS datastructure in the case of large matrix groups since finding a BSGS datastructure of practical size in such groups is often impossible.

The LMG family of functions assume that a special type of composition series is used so the first step is to transform the CT returned by the O'Brien et al package into one corresponding to the special composition series. All LMG functions assume that the CT is in this form. Structural information such as maximal subgroups for G can now be computed by first finding their preimages in the quotient group G/R, where R is the soluble radical of G, and then extending these results to G by lifting through successive sections of R. At present the LMG intrinsics have the prefix 'LMG' added to the intrinsic name. It is hoped to merge the LMG functionality with the standard functions in the near future.

New Features:

- The character table algorithm of Bill Unger (based on Brauer's Theorem) has been generalised to work for matrix groups having a CT datastructure (intrinsic `LMGCharacterTable`).
- The normal subgroups of an LMG group may be computed using the intrinsic `LMGNormalSubgroups`.
- The subgroups having index less than a small bound may be computed using the intrinsic `LMGLowIndexSubgroups`.
- The action of an LMG group on the cosets of some subgroup may be found using the intrinsics `LMGCosetAction`, `LMGCosetImage` and `LMGRightTransversal`.

10.6 Permutation Groups

New Features:

- The fundamental routines for computing the normal subgroup structure of a permutation group have been extended to apply to all permutation groups possible in Magma. Previous versions were limited to degree 10^7. This includes computations of socle, chief series, composition series, chief factors, composition factors, and simplicity testing.
- The function `DoubleCosetCanonical` which computes a canonical representative of a double coset has been included. Given a covering group, the two subgroups, and a double coset representative, it uses backtrack search to compute the minimal base image of an element of the double coset. This may be used to test membership in or equality of double cosets.

Bug Fixes:

- A crash when computing the socle of a permutation group has been fixed. (V2.20-9)
- A crash when computing the conjugacy classes of a permutation group has been fixed. (V2.20-7)
- A crash when computing the derived series of a permutation group has been fixed. (V2.20-7)
- A bug where using `NormalSubgroups` with the `IsTransitive` flag set returned some non-transitive subgroups has been fixed. (V2.20-6)

10.7 Automorphism Groups

New Features:

- It is now possible to obtain a presentation of the automorphism group of a free group using the `FPGroup` function. An improved method is used for computing the inverse of an automorphism of these groups.

10.8 Databases of Groups

New Features:

– The transitive group database has been extended by Derek Holt to include the transitive groups having degrees $33 - 47$ inclusive.

11 Lattices and Quadratic Forms

11.1 Lattices

New Features:

- Magma should now use Damien Stehle's LLL implementation in (almost) all cases. Previously, it was being used for all applications of LLL to a matrix, but not for LLL when applied to a lattice given by real input rather than exact input. Reported by M. Kirschmer. (V2.20-4)
- The `Coordinates` intrinsic can now (sometimes) work over real fields. (V2.20-9)

Changes and Removals:

- The `LatticeWithBasis` intrinsic for a matrix over the reals checks that the input has full rank using LLL now (previously the check was done by calculating an echelon form). This means now almost any input is accepted, as the linear dependence must be exact for the check to fail. (V2.20-6)

Bug Fixes:

- A problem with `ClosestVectors` has been fixed. Reported by M. Kirschmer and G. Nebe. (V2.20-8)
- Some discrepancies with varargs to `LLL` have been fixed. In particular, the Lovasz condition was not always the default, and `Delta` was ignored for `LLL` of a lattice. (V2.20-8)
- A memory leak with the `Coordinates` function has been fixed. (V2.20-10)

11.2 Quadratic Forms

Bug Fixes:

- A fix was made to the intrinsic `WittInvariants` so that the set of bad primes is now consistent with the Handbook description. (V2.20-4)
- A bug with the improper use of C-integers for an indefinite Gram matrix has been fixed. Noted by T. A. Fisher. (V2.20-9)
- The code for `IsotropicSubspace` has been slightly reordered, to apply `LLLGram` after each prime (rather than just at the end). The previous code would double the size of entries at each step, which could lead to coefficient blow-up when the determinant had many prime factors. Noted by T. A. Fisher. (V2.20-9)
- The changes (efficiency improvements) in the ordering of reduction and minimisation steps in the previous patch release failed to properly account for the unimodular case (there are thus no primes to minimise, and hence reduction is not done). Reported by T. A. Fisher. (V2.20-10)

11.3 Binary Quadratic Forms

Changes and Removals:

- For positive discriminants (with fundamental unit of norm 1), the behaviour of `ReducedForms` and `ReducedOrbits` was illogical and has been changed: now every form is equivalent to a reduced form. Note: confusion arises from the fact that the `ClassGroup` of binary quadratic forms in Magma is not the most natural one in the context of forms, and this was never documented.

12 Linear Algebra and Module Theory

12.1 Matrices

New Features:

- An analogue for the LLL lattice basis reduction algorithm is now available for matrices with entries in $K[x]$, for K a field (intrinsic function `LLL` applied to a matrix with entries in $K[x]$). The algorithm uses asymptotically-fast techniques when the degree of the polynomials in the input matrix is large and/or the dimension of the matrix is large.
- A new asymptotically-fast algorithm for computing the Hermite Normal Form (HNF) of a dense matrix over $K[x]$ (K a field) has been developed, which builds on the above LLL algorithm for matrices over $K[x]$. This algorithm is dramatically faster than the previous algorithm for uniform dense matrices, in general. For example, computing the HNF with transformation matrix of a dense 100×100 matrix over GF(32003)$[x]$, where the input entries are random polynomials of degree 3, takes 0.6 seconds, which is 32 times faster than the previous algorithm.
- New asymptotically-fast algorithms for computing the determinant, rank, nullspace, inverse, or Smith Normal Form of a matrix over $K[x]$ have been developed, based on the above HNF algorithm.
- A new asymptotically-fast algorithm for computing the echelon form of a matrix over the rational function field $K(x)$ (K a field) has been developed, which again builds on the above LLL algorithm for matrices over $K[x]$. This algorithm is even more dramatically faster than the previous algorithm in general. For example, computing the inverse of a dense 100×100 matrix over GF(32003)(t), where the input entries are random polynomials of degree 3, takes 3.7 seconds, which is 205 times faster than the previous algorithm.
- New asymptotically-fast algorithms for computing the determinant, rank, nullspace, or inverse of a matrix over $K(t)$ have been developed, based on the above echelon form algorithm.
- A multi-threaded parallel version of matrix multiplication has been developed for matrices over GF(2). If the user first calls the procedure:

  ```
  SetNthreads(k);
  ```

  then k threads will be used when multiplying sufficiently large matrices over GF(2). This often leads to a speedup in wall-clock time close to a factor of k. Multiplication of large matrices over GF(2^k) for $k > 1$ often maps to the GF(2) algorithm, so a similar speedup will be present in such cases.
- The asymptotically-fast Keller-Gehrig algorithm for computing the minimal polynomial of a matrix over a field has been implemented. The algorithm can be selected by setting the parameter `Al` to the string `"KellerGehrig"` for the function `MinimalPolynomial`. The algorithm uses $O(\log_2(N))$ matrix multiplications and one nullspace computation, where N is the dimension of the matrix, so for large N and for reasonably large base finite fields, the algorithm may be rather faster than the default algorithm, especially if a GPU is present.
- The algorithm for computing the Hermite Normal Form of matrices with some sparsity has been sped up, particularly in the case where there are repeated elementary divisors.
- Compatibility between non-square matrices has been improved, in particular between `ModMatRngElt` and `ModMatFldElt` types.

– The mutation operators `+:=` and `-:=` have been extended to work when the objects on both sides are compatible matrices which may have different types. Omission reported by A. Maurischat. (V2.20-4)

Bug Fixes:

– Some incorrect results involving matrix multiplication over large non-prime finite fields (with a modular algorithm) have been fixed. Reported by B. Tsaban.

– A crash involving indexing of a matrix with very many rows has been fixed.

– A problem with matrix multiplication over rational function fields (where entries were not standardized so wrong results could arise occasionally) has been fixed. Reported by E. Schost. (V2.20-3).

– A hang in the p-adic integer nullspace algorithm has been fixed. Reported by T. Fisher. (V2.20-9)

12.2 Modules over Dedekind Domains

New Features:

– More functionality has been provided for pseudo matrices over algebraic function fields. It is now possible to construct a `PseudoMatrix` from a sequence of `RngFunOrdIdl` and a matrix over a function field, multiply such a pseudo matrix by an `RngFunOrdIdl` and retrieve the `Matrix` and `CoefficientIdeals` of these pseudo matrices. (V2.20-2)

12.3 Numerical Linear Algebra

New Features:

– Magma now has methods based on Householder reflections (RQ decompositions) for various functions involving linear algebra over a real or complex field. The user functionality now includes `RQDecomposition`, `QLDecomposition`, `NumericalKernel`, `NumericalRank`, `NumericalSolution`, and `NumericalPseudoinverse`. The main functions for determinant and inverse for square matrices use the new code internally. (V2.20-6)

Bug Fixes:

– The `NumericalEigenvectors` intrinsic has been fixed to deal with input that has columns that are zero. Reported by A. Kumar. (V2.20-6)

– The `Cholesky` decomposition of a matrix which has different precision than the default real field is now computed to the proper precision. (V2.20-6)

– A problem with randomisation in `QLCDecomposition` has been fixed. This should lead to superior performance with similar functions. (V2.20-8)

13 Linear Associative Algebras

13.1 Associative Algebras

New Features:

- A `MaximalOrder` containing an order of an associative algebra can now be computed.
- An associative algebra can now constructed from an algebra of type `AlgGrp` or `AlgMat` using `AssociativeAlgebra`.

13.2 Clifford Algebras

Clifford algebras are structure constant algebras, implemented using the quadratic space machinery in Magma.

New Features:

- New intrinsics have been written for Clifford algebras and the Handbook has been edited to include descriptions of several previously undocumented intrinsics.
- If C is the Clifford algebra defined by a quadratic form Q on a vector space V of dimension n, the elements of C can be represented as sums of monomials in the basis elements of V. Intrinsics are provided to convert between this representation and the internal representation as elements of a vector space of dimension 2^n and also to print an element as a (non-commutative) polynomial in the basis vectors.
- Given a Clifford algebra, there is now code to return the centre, the even subalgebra and the homogeneous component of a given degree.
- The code connecting Clifford algebras, orthogonal groups and spin groups has been improved.
  - If g is an invertible element of a Clifford algebra and if g preserves the quadratic space V, then `VectorAction(g)` returns the matrix of g acting on V.
  - The Handbook gives examples of the connection between the vector action, Siegel transformations and spin groups.

13.3 Finitely Presented Associative Algebras

Bug Fixes:

- A crash in `ChangeRing` for finitely-presented algebras has been fixed. Reported by F. Lunnon. (V2.20-10)

14 Representation Theory

14.1 Modules over Algebras

Bug Fixes:

– A bug in `IsIsomorphic` for R-modules in characteristic zero has been fixed. Reported by U. Thiel. (V2.20-6)

14.2 Character Theory

New Features:

– A version of Bill Unger's algorithm for computing the table of irreducible complex characters has been implemented for large degree matrix groups. At present there are some limitations but it is now possible to compute character tables for matrix groups over $GF(q)$ having substantial degree. The intrinsic is `LMGCharacterTable`.

– The `Symmetrization` of a character can now be computed for larger partitions than previously. Furthermore, the previous code had some errors for partitions of size 6. (V2.20-8) The associated `OrthogonalComponent` and `SymplecticComponent` intrinsics were similarly buggy, and their usage should be replaced by the new `OrthogonalSymmetrization` and `SymplecticSymmetrization`.

– Intrinsics for the recognition of quasi-elementary and $\mathbf{Q}$-groups, and virtual permutation characters, have been added.

Bug Fixes:

– The printing of character tables has been fixed to print integers of magnitude $\geq 2^{30}$ correctly. In the past such integers were incorrectly displayed when printed as part of a character table. The internal representation was correct, as was the representation of any single character value and character, the only error was in printing out a full table. (V2.20-5)

15 System

15.1 GPU support

New Features:

– The procedure `SetGPU` sets whether Magma should use NVIDIA GPUs via CUDA when present. This is only relevant to a CUDA-enabled executable (downloaded as `magma.cuda.exe`) and is `true` by default in that case (so a GPU is used by default); for a non-CUDA-enabled executable, the procedure has no effect. Currently, a GPU is exploited in matrix multiplication over GF(2) and small prime finite fields and consequently anything which depends on such multiplication, such as the dense F_4 Gröbner basis algorithm over such fields.