Release Notes (Version 2) for Magma V2.25

20 January 2020

This document provides a terse summary of the new features included in Magma versions V2.25-3 onwards (released 20 January 2020). It is in two parts, the first part being a concise listing of the more noteworthy new features while the much longer second part gives details of all new features together with other changes that appear in V2.25 for the first time. The second part also notes a few of the significant bugs that have been fixed in the past 12 months. It is far from complete and a reader wishing to get a full listing of bugs fixed in the past 12 months should consult the patch release change log for V2.24-x.

Part 1: Summary of New Features in Magma V2.25

Language and System

- **New Type for I/O Objects**
  - **Important: All I/O objects (currently files, pipes, and sockets) now have type `IO`. Existing package code that uses the old `File` type will need to be changed.**
- *Serialisation*
  - It is now possible to write or read some basic types of Magma objects using a binary format instead of a textual representation. The relevant intrinsics are `WriteObject` and `ReadObject`. These allow Magma objects to be directly written to or read from an I/O object (file, pipe, or socket). This has advantages of using less space for large objects, of not requiring parsing the text format, and of avoiding issues of partial data — an object is either read in its entirety or not at all.

    In this initial release the supported objects that can be transmitted include integers, rationals, reals, complexes, booleans, strings, finite field elements, polynomials (univariate and multivariate), sequences, sets, indexed sets, multisets, tuples, lists, records, matrices, vectors, lattices, permutations, number field elements, and all the associated parent structures of these. Support for the transmission of more types will be added in patch releases as part of an ongoing project.

A separate document detailing the object transmission format and protocol is also provided, and will be updated as new types are supported. The hope is that this will facilitate interaction between Magma and other (possibly bespoke) computational software.

- *Asynchronous I/O*
  - Asynchronous I/O facilities have been added which enable I/O requests to be queued without needing to wait for them to complete. This greatly simplifies writing distributed parallel code, particularly in conjunction with the ability to write or read entire objects as mentioned above.
- *Dual Iterators*
  - Iteration through aggregate types that have an indexing operation now also supports a dyadic form which allows both the index and value to be specified in the iteration. This allows a much more natural way of expressing the loop when both index and value are wanted. Aggregates that provide dual iterators are sequences, indexed sets, associative arrays, tuples, lists, and multisets (where the index and value are taken to be the element in the base set and the multiplicity, respectively).
- *Timing With Multi-threaded Algorithms*
  - The behaviour of the `time` and `vtime` statements has been modified in the case that a multi-threaded algorithm is used and a new intrinsic function `Time` has been added which is useful if a multi-threaded algorithm is used; see the details in Subsec. 1.2 below.

The Magma Handbook

A number of changes have been made to the Handbook for Magma V2.25. The intention is to make the most sought-after information easier to locate and information generally more accessible.

- A new introductory chapter "Introduction to Finitely-Presented Groups" has been prepared. This is a much shorter (50 pages) treatment of the material in the two chapters covering finitely-presented groups. The introductory chapter presents basic descriptions of all the main intrinsics but omits those that are only occasionally used.

- The existing material in the chapter "Finitely Presented Groups" has been reorganised in a way that hopefully makes things easier to find. Also, the material from "Finitely Presented Groups: Advanced" has been integrated into the revised chapter. The general idea is that the Introduction will provide most necessary information while this chapter will be the reference when deeper information about a topic is needed.

- A new chapter "Free Groups" has been created, bringing together the information for free groups that was formerly spread across the two finitely-presented group chapters.

- The material on hyperbolic groups has been moved to the chapter "Automatic Groups".

- A new chapter called "Modules over Algebras and Group Representations" has been formed from the material in chapters "Modules over Algebras" and "$K[G]$-Modules and Group Representations". The material has been extensively revised to bring it up to date and also to make it slightly more accessible. In particular, the intrinsics that apply to modules over number fields have been identified. Further improvements to this chapter are planned.

Algebraic Geometry

- *Curves*
  - A new intrinsic is provided which computes the *delta adjustment* for a singular point on a curve. i.e., the amount that the singularity contributes to the excess of the arithmetic genus of the singular model of the curve over the genus of the normalisation. This is used in the implementation of an alternative algorithm for computing the genus of a curve which is sometimes much faster than the standard genus algorithm.

- *Surfaces*
  - A significant amount of functionality that was either available only for surfaces with simple singularities or, in some cases, with general singularities but lying in $\mathbf{P}^3$ has been extended to all surfaces with general singularities in $\mathbf{P}^3$ or arbitrary point-singularities in $\mathbf{P}^n$, $n \geq 4$. This includes a number of invariants such as `KodairaDimension` and `KodairaEnriquesType`.
  - For ordinary projective surfaces with only point singularities, the calculation of the intersection numbers of *strict transforms* of effective divisors on the blow-up desingularisation is now available.
  - The computation of explicit bases of holomorphic 2-forms or n-pluricanonical forms has been implemented. For $n \geq 2$ this works for any ordinary projective surface with only point singularities. Additionally, general adjoint maps given by `HomAdjoints` and `CanonicalIntersection` for the strict transforms of divisors has been extended from hypersurfaces to general ordinary projective surfaces (with the singularity restriction). The divisor of a "base" meromorphic 2-form is also computed.
  - The computation of all (-1)-curves (including Galois-conjugate sets of such) for ordinary projective surfaces with only point singularities of Kodaira dimension ≥ 0 (i.e., not rational or birationally ruled) has been implemented.
  - Introduction of a *dual resolution graph* to make explicit the configuration of irreducible blow-up components over a singular point in the blow-up desingularisation. This also provides additional technical information about the blow-up components and their intersections and there is a minimal version to compute the graph of the minimal desingularisation (when (-1)-curves above the singularity are removed). There are also intrinsics to count the number of points or compute the zeta-function of the reduced divisor lying over the singular point when the surface is defined over a finite field. These are very useful for point-counting on the entire desingularised surface.

– A package of (initial) functionality for working with degree-2 K3 surfaces, given as double covers of the projective plane ramified over a sextic in $\mathbf{P}(1,1,1,3)$ weighted projective space is included. This concentrates on finding divisors, Neron–Severi group computations and finding elliptic fibrations of the surface. It has the following functionality:
  * Intrinsics for finding (-2)-curves on the surface by computing the lines and conics in the plane that split in the cover.
  * An intrinsic to compute the full intersection matrix of the strict transforms of a given collection of effective divisors on the surface along with the blow-up (-2)-curves over the (simple) singular points of the ramification sextic.
  * For elliptic fibrations there is a suite of intrinsics to compute the following:
    1. Configuration data for all distinct elliptic fibrations with a bad fibre supported on a given set of (-2)-curves (including blow-up ones).
    2. The explicit fibration map for any of these configurations.
    3. A model of the genus 1 generic fibre of a given fibration along with the points on it corresponding to given sections.

Arithmetic Geometry

- *Elliptic Curves Over* **Q**

  - The Cremona database of all elliptic curves of small conductor has been updated to include all conductors up to 499,999. In this update curves with conductors in the range 400000 to 500000 have been added thereby introducing an additional 581,056 curves in 423,257 isogeny classes.

- *Hyperelliptic Curves and Curves of Genus* 2

  A major upgrade of the packages for hyperelliptic curves and curves of genus 2 has been undertaken by M. Stoll. It includes:-

  - The code for computing heights of points on Kummer surfaces and on Jacobians of genus 2 curves, both over **Q**, has been completely overhauled.
  - Functionality is provided for computing *double Richelot isogenies* over **Q** and more general isogenies of degree a power of 2. These isogenies are defined over the rationals but are composites of two Richelot isogenies defined over a quadratic field.
  - Principally polarised abelian varieties of dimension two which are isogenous over the rationals to a genus 2 Jacobian over the rationals by an isogeny of degree a power of two can be computed.
  - For the first time the Mordell-Weil group of a hyperelliptic Jacobian of genus 2 over the rationals can be computed.
  - Mmore generally, the Mordell-Weil group of a hyperelliptic Jacobian of genus 1 or 2 over, respectively, a number field or the rationals, can be computed.

- *Binary and Ternary Forms*

  - Tools for the minimization and reduction of binary forms defined over the rationals or integers have been provided by M. Stoll.
  - A package for the minimization and reduction of ternary forms over the rationals, recently developed by A. S. Elsenhans and M. Stoll, has been included in Magma.

Arithmetic Fields

- *Algebraic Number and Function Fields*

  - In 1999, J. Montes introduced a new computational representation, the *OM representation*, for prime ideals in Dedekind domains. This is the basis of a new approach for computing with fractional ideals in Dedekind domains. This approach has been applied to ideal computations in both number fields and algebraic function fields. This can lead to dramatic speeds ups in the computation of maximal orders and the factorisation of ideals in number fields. In the case of function fields it leads to speed-ups, for example, in the computation of genus, the (finite) maximal order and Riemann-Roch spaces. A Magma package implementing the Montes approach for both number fields and function fields has been developed by Jens-Dietrich Bauch and parts of it are exported for the first time in this release.

- *Galois Groups*

  - A major improvement has been made to the computation of Galois groups by A. S. Elsenhans in the case in which we have a large degree number field having a medium size subfield. The improvement applies to fields of degree greater than 30. For example, computing the Galois group of degree 96 fields having a subfield of degree 16 is possible for the first time.

Basic Rings and Fields

- Integer Ring
  - In 64-bit mode, a large integer may now have up to 2^{69} bits (the previous limit was up to 2^{37} bits). This also allows multiplication of higher degree univariate polynomials (over the integers and integer residue rings, etc.) than was previously possible.
- Polynomial Rings
  - The general factorisation algorithm for univariate polynomials over $\mathbf{Z}$ or $\mathbf{Q}$ has been greatly sped up for several types of input, by optimising various stages of the algorithm.
  - The Trager factorisation algorithm for univariate polynomials over an algebraic number field $K = \mathbf{Q}(\alpha)$ has been greatly sped up for several types of input, by optimising various stages of the algorithm.
  - The Brent-Kung algorithm for modular evaluation has been significantly sped up for polynomials defined over large prime finite fields. As a result, the algorithm for factoring polynomials over large prime finite fields is now significantly faster.
  - The modular GCD algorithm for univariate polynomials over an algebraic number field $K = \mathbf{Q}(\alpha)$ greatly sped up for several types of input.
  - The squarefree factorisation algorithm has been improved for larger characteristic p fields. This can also speed up the initial squarefree phase of the general factorisation algorithm over such fields.

Commutative Algebra

- *Gröbner Bases*
  - The dense F_4 Gröbner basis algorithm now has multithreading support (based on multithreading support for the underlying linear algebra) for dense ideals defined over the field GF(p) for prime $p \leq \lfloor 2^{23.5} \rfloor = 11863283$.
- *Ideal Arithmetic*
  - The algorithm to compute the radical of a positive-dimensional ideal has been greatly improved for certain classes of inputs.

Complex Manifolds

- *Riemann Surfaces*

  - A package for Riemann surfaces defined by either an affine equation $f(x, y) = 0$ or an equation of a superelliptic curve has been implemented by C. Neurohr.
  - Basic information such as genus, big period matrix, small period matrix, discriminant points, branch points, ramification points, singular points and fundamental group can be computed.
  - The main focus is on approximating period matrices and the Abel-Jacobi map of divisors of the Riemann surface X to the Jacobian of X using numerical integration. Working with a precision of one hundred decimal digits is practical in the general case and several thousand digits in the superelliptic case.

Group Theory

- *Finitely-Presented Groups*

  - An intrinsic has been implemented for constructing the automorphism group of a free group by Derek Holt and others.
  - The test for a finitely presented group being hyperbolic has been expanded to return and apply a Dehn algorithm.
  - The code that searches for simple group quotients of a finitely presented group has been upgraded to search for simple group quotients of order up to 10^{10} (up from 10^9).
  - The code for finding normal subgroups of small index in a finitely presented group has been upgraded from an index limit of 50,000 to an index limit of 100,000.

- *Permutation Groups*

  - A canonical form for permutation groups is now available.
  - The database of transitive groups has been extended to include the 195,826,352 transitive groups of degree 48 which were recently constructed by Derek Holt. So the transitive group database now includes the transitive groups for all degrees less than 49.

- *Matrix Groups*

  - A substantial update to the Composition Tree package is included. This incorporates constructive recognition code for the families of finite exceptional groups of rank at least 2, and so it also extends the range of various LMG functions

for large matrix groups to include groups with such composition factors. The Composition Tree code is developed and maintained by Henrik Baarnhielm and Eamonn O'Brien.

- The short presentations of Leedham-Green and O'Brien for classical groups on their standard generators are now available, and are used in the verification of composition trees for matrix groups.
- A package developed by Bettina Eick, Tommy Hofmann and Eamonn O'Brien for testing conjugacy of elements in $GL(n, \mathbf{Z})$ has been developed. The centraliser of an element of $GL(n, \mathbf{Z})$ can also be computed.
- The machinery for computing complements of a normal subgroup in a permutation group has been extended to complements of a normal subgroup in a finite matrix group.
- An efficient algorithm for computing the order of a matrix having entries in the ring $\mathbf{Z}/n\mathbf{Z}$, for any positive integer n, has been implemented. (Previously a naive brute force search algorithm was used.)

- *Classical Groups*
  - Maximal subgroups and automorphism groups have been installed for the following classical groups and their almost simple extensions: $L_8(3)$, $L_9(3)$, $U_5(4)$, $U_6(3)$, $S_{14}(2)$, $O^+_{14}(2)$ and $O^-_{14}(2)$.
  - The short presentations of Leedham-Green and O'Brien for classical groups on their standard generators are now generally available.

- *Exceptional Groups*
  - Constructive recognition for the families of final exceptional groups of rank at least 2 is now available. Standard generator are constructed using the algorithms of Liebeck and O'Brien; algorithms of Cohen, Murray and Taylor are used to write elements of the exceptional group as words in these standard generators; presentations on these standard generators are also available. The code was prepared by Eamonn O'Brien and Don Taylor. Equivalent machinery for the remaining families of exceptional groups was already available.
  - Maximal subgroups and automorphism groups have been installed for the exceptional groups $F_4(2)$ and Ree(27) and their almost simple extensions.

- *Sporadic Groups*
  - Maximal subgroups and automorphism groups have been installed for the sporadic simple groups HN and Fi_{23} and their almost simple extensions.

Lattices and Quadratic Forms

- *Lattices*
  - Vector enumeration has been parallelised using *pthreads*. The speed-up achieved is close to linear in the number of cores used.
  - Various lattice functions involving integral lattices (such as enumeration) now use information about the evenness of a lattice when applicable.

Lie Theory

The following functionality has been added for simply connected finite groups of simple Lie type by Don Taylor. In particular, the row reduction algorithm has been implemented for twisted groups of types ${}^3\mathrm{D}_4$ and ${}^2\mathrm{E}_6$.

- *Curtis–Steinberg–Tits Presentations*
  - Matrix generators (in a given highest weight representation), which satisfy the Curtis–Steinberg–Tits (CST) relations are available for finite untwisted groups and twisted groups of types ${}^2\mathrm{A}_n$ (n odd), ${}^2\mathrm{D}_n$, ${}^3\mathrm{D}_4$ and ${}^2\mathrm{E}_6$.
  - It can be verified whether a collection of matrices satisfies the CST relations for a specified group of Lie type.
  - Given matrices which satisfy the CST relations for a simply connected finite group G of Lie type, the homomorphism from G to the matrix group can be constructed. This extends existing code which was available only for a limited number of representations.
- *Representation Theory*
  - The highest weight and a maximal vector can be computed for each highest weight representation.
  - The contravariant form of a highest weight representation of a finite group of Lie type is available.
  - The Chevalley normal form of a matrix in the image of a representation of a finite group of Lie type can now be found for all twisted types (except type ${}^2\mathrm{A}_{2n}$) and all irreducible representations. This machinery extends existing "row reduction" functionality for untwisted groups and can be used to determine whether a matrix is in the image of the representation.
  - Combined with group recognition work of Eamonn O'Brien, the row reduction machinery can be used to construct a (projective) isomorphism from a simple matrix group of Lie type to a standard copy.

Linear Algebra

- *Linear Algebra Over Finite Fields*
  - Multithreading is now supported for the multiplication of large matrices over the field GF(p) for prime $p \leq \lfloor 2^{23.5} \rfloor = 11863283$.
  - The base algorithm for matrix multiplication in large dimension has been significantly improved for matrices over GF(2^k) for $k = 2, 3, 4$.
  - A major improvement in both time and memory usage has been achieved for multiplication of large matrices over GF(q) where $q = p^k$, $p > 2, k > 1$, and $q < 256$.
  - Matrix transpose over GF(q) for $q = 3, 4, 5, 7, 8$ has been significantly improved (for all matrix sizes).
  - Matrix multiplication over GF(p) for $p = 3, 5, 7$ has been significantly improved (for all matrix sizes) in the AVX version.
- *Linear Algebra Over General Rings*
  - The recursive echelon algorithm has been improved: (1) in the case where a transformation matrix is required (particularly when the number of rows is less than the number of columns); and (2) in the case where the base ring is GF(q) for $q \leq 7$. This also yields improvements to the many algorithms which depend on this, including the computation of the inverse of a matrix.
  - The modular algorithm for multiplication of matrices with large integer entries (including integers in a large residue class ring) has been significantly sped up, particularly in the case that the size of the integers is large.
  - Improvements have been made in the multiplication, echelonisation and computation of the the nullspace for matrices with integer or rational entries.
- *Sparse Matrices*
  - The main sparse Gaussian elimination algorithm has been greatly improved for very large sparse matrices, in reduction of both memory usage and time taken. For sparse matrices with at least one dimension over a million, the improvement can be quite dramatic.
  - Even greater reduction in memory and CPU time usage in the main sparse Gaussian elimination algorithm has been achieved for very large sparse matrices over GF(2).

Modular Forms

- *Newforms*
  - Major improvements have been made to the computation of newforms of cuspidal subspaces. For example, compared with Magma V2.24, the evaluation of the expression `Newforms(CuspForms(2275, 4))` is about 50 times faster.
  - Even greater improvements have been made for computing newforms of cuspidal subspaces twisted by a Dirichlet character. For example, compared with V2.24, the statement

    ```
    Newforms(CuspForms(G.1^2, 2)) where G := FullDirichletGroup(N);
    ```

    is now about 670 times faster for $N = 139$, and about 1960 times faster for $N = 163$.
  - The setting up of an absolute number field and the application of the appropriate isomorphism (when needed) to compute q-expansions has been greatly improved. For example, the printing of the newforms which are returned by

    ```
    Newforms(CuspForms(FullDirichletGroup(81).1^2, 2));
    ```

    (which involves the computation of q-expansions) is now about 110 times faster than for V2.24.

Representation Theory

- *$K[G]$-Modules, K a Finite Field*

  - The general modular Meataxe is now significantly faster in general. In particular, it is much faster in the case in which the resulting quotient module has large dimension.
  - Independently of the Meataxe improvements, the modular spin algorithm has been sped up.
  - A more sophisticated algorithm for computing the Brauer character of a $K[G]$-module has been developed with the result that the computation for modules having large dimensions is now a great deal faster.

- *$K[G]$-Modules, K a Number Field*

  - A large number of $K[G]$-module intrinsics can now be applied to $K[G]$-modules, where K is the rational field or a number field. Routine computation is practical for modules over $\mathbf{Q}$ for dimensions at least up to a thousand. In the case of modules over low degree number fields, computation is practical up to dimension several hundred.
  - A rational Meataxe and a number field Meataxe are available to split modules. So all of the operations that depend upon the Meataxe such as endomorphism rings of modules and isomorphism of modules are available.
  - A module can be tested for decomposability and its decomposition found.
  - Irreducible modules can be computed given the character of the module.

- *$(K[G],\ K[H])$-bimodules*

  A basic facility for working with bimodules developed by Derek Holt and Allan Steel is included in this release. In the Magma implementation, $(K[G],\ K[H])$-bimodules are regarded as being equivalent to standard right $K[G \times H]$-modules. Most of the functionality for standard right $K[G]$-modules, such as forming sub- and quotient modules, defining homomorphisms, etc, should also work for $(K[G],\ K[H])$-modules, where results are calculated using the equivalent $K[G \times H]$-module. A tensor product intrinsic is provided as tensor products dont work in this approach to computing with bimodules.

- *Brauer Characters*

  A suite of intrinsics for computing with Brauer characters has been installed.

Part 2: Details of Changes and New Features in Magma V2.25

1 Language and System Features

1.1 I/O

New Features:

- **Important: All I/O objects (currently files, pipes, and sockets) now have type `IO`. Existing package code that uses the old `File` type will need to be changed.**
- The intrinsics `WriteObject` and `ReadObject` have been added. These allow Magma objects to be directly written to or read from an I/O object (file, pipe, or socket) using a binary format instead of a textual representation. This has advantages of using less space for large objects, of not requiring parsing the text format, and of avoiding issues of partial data — an object is either read in its entirety or not at all.

  In this initial release the supported objects that can be transmitted include integers, rationals, reals, complexes, booleans, strings, finite field elements, polynomials (univariate and multivariate), sequences, sets, indexed sets, multisets, tuples, lists, records, matrices, vectors, lattices, permutations, number field elements, and all the associated parent structures of these. Support for transmission of more types will be added in patch releases as part of an ongoing project.

  A separate document detailing the object transmission format and protocol is also provided, and will be updated as new types are supported. The hope is that this will facilitate interaction between Magma and other (possibly bespoke) computational software.
- Asynchronous I/O has been introduced, allowing I/O operations to be queued for later completion. This should greatly simplify the process of using Magma for distributed parallelism. Each of the major I/O operations (`Read`, `ReadBytes`, `ReadObject`, `Write`, `WriteBytes`, and `WriteObject`) also has an associated asynchronous version (`AsyncRead`, `AsyncReadBytes`, etc.).

  Asynchronous I/O operations always return immediately. Asynchronous reads must be followed at a later point by a corresponding synchronous read to retrieve the data once it has been determined to be available. The `WaitForIO` intrinsic can be used to test for this. More details are in the Handbook chapter on Input and Output.
- New function `Pipe(C, S)` taking parallel sequences C and S, so that for each i, the shell command $C[i]$ is executed with input string $S[i]$ (all in parallel) and the sequence of corresponding output strings (matching each command) is returned.

Changes and Removals:

- The type `File` used for files and pipes has been changed to `IO`.

1.2 Magma Language

New Features:

– Iteration through aggregate types that have an indexing operation now also supports a dyadic form which allows both the index and value to be specified in the iteration. Aggregates that provide these "dual iterators" are sequences, indexed sets, associative arrays, tuples, lists, and multisets. (Although multisets do not have an indexing operation, they do have a natural dyadic view linking an element of the base set with its multiplicity.)

Dual iteration is indicated using the arrow operator `->`. Some trivial examples of its use are below.

```
> P := PrimesUpTo(10);
> for i->p in P do
>   printf "Prime %o is %o\n", i, p;
> end for;
Prime 1 is 2
Prime 2 is 3
Prime 3 is 5
Prime 4 is 7

> M := Multiset(Eltseq("hello world"));
> for c->count in M do
>   if count eq 1 then
>     isare := "is"; plural := "";
>   else
>     isare := "are"; plural := "s";
>   end if;
>   printf "There %o %o '%o'%o\n", isare, count, c, plural;
> end for;
There is 1 ' '
There is 1 'e'
There are 3 'l's
There is 1 'w'
There is 1 'h'
There are 2 'o's
There is 1 'd'
There is 1 'r'

> S := [];
> for k in [1..3] do S[k^2] := k; end for;
> S;
[ 1, undef, undef, 2, undef, undef, undef, undef, 3 ]
> valid_indices := [ i : i->_ in S ];
> valid_indices;
[ 1, 4, 9 ]
```

– The `time` statement has been modified so that if a multi-threaded algorithm is used within the executed statement and the real time r taken is less than the CPU time c taken, then the time printed is r, and the tag `[r]` is appended (to indicate that the time printed is a real time); otherwise the CPU time c is printed. So there is no change to the behaviour if a multi-threaded algorithm is not used (since the CPU time is always printed as previously). So one can know whether a multi-threaded algorithm has been non-trivially used within the executed statement: that is the case if

and only if the tag `[r]` is printed. The verbose time statement `vtime` now has the same behaviour too.

Similarly, a new function `Time` has also been added, which is used in a similar way to `Cputime` and `Realtime`: by first setting a variable such as T to `Time()`, then after a series of statements, one can print the value `Time(T)` (which is always a string) to give the lapsed time since T was assigned. Just as for the `time` statement above, the tag `[r]` is added if and only if the real time is used in the second call to `Time`.

A simple example of the behaviour is the following:

```
> X := Random(MatrixRing(GF(5), 10000));
> time P := X*X; // single thread, normal CPU time
Time: 4.060
> T := Time(); P := X*X; Time(T); // similar with Time() function
4.060

> SetNthreads(8);
> time P := X*X; // multi-threads, so real time shown by [r]
Time: 1.050[r]
> T := Time(); P := X*X; Time(T); // similar with Time() function
1.050[r]
```

– The `exit` (or `quit`) statement can now take an argument giving the desired exit status.

Bug Fixes:

– A crash in `AttachSpec` has been fixed. (V2.24-4)

– A crash in `save/restore` has been fixed. (V2.24-7)

– A memory management problem to do with multisets has been fixed. (V2.24-9)

2 Magma Handbook

A considerable number of changes have been made to the Handbook for Magma V2.25. Some of these are steps towards making some of the material more accessible.

2.1 Algebraic Geometry Chapters

New Features:

– A new chapter "Riemann Surfaces" documents the intrinsics for Riemann surfaces.

– The chapter on Algebraic Surfaces has undergone considerable expansion.

2.2 Basic Rings and Fields Chapters

New Features:

– Documentation of the numerical integration intrinsics implemented by Christian Neurohr has been included in chapter "Real and Complex Fields".

2.3 Group Theory Chapters

New Features:

- A major reorganisation of the material on finitely-presented groups (fp-groups) has been undertaken. Up until the present most of the material has been located in two chapters "Finitely-Presented Groups" (FP1) and "Finitely-Presented Groups: Advanced" (FP2). The combined length was around 180 pages and it had become difficult to locate the relevant intrinsics for a given problem. In order to make it easier for users major changes have been made which are outlined below.
- A new chapter ("Free Groups") bringing together the intrinsics for free groups that was formerly spread across the chapters FP1 and FP2 has been created.
- The material on hyperbolic groups that was formerly in chapter FP1 has been moved to the chapter on automatic groups ("Automatic Groups").
- A new introductory chapter on fp-groups ("Introduction to Finitely-Presented Groups") has been prepared. This is a much shorter (50 pages) treatment of the material in chapters FP1 and FP2. The introductory chapter presents basic descriptions of all the main intrinsics but omits many of the more specialised intrinsics.
- The existing material in FP1 has been reorganised in a way that hopefully makes things easier to find. Also the material from FP2 has been integrated into the revised FP1. The general idea is that the Introduction will provide most necessary information while FP1 is referred to when further information on a topic is needed.
- With its information having been moved into FP1, chapter FP2 has been deleted.

2.4 Representation Theory Chapters

Changes:

- In response to user feedback the material in the chapter "Modules over an Algebra" has been integrated into the chapter "$K[G]$-Modules and Group Representations" and this combined chapter has been renamed to "Modules over Algebras and Group Representations". The former chapter "Modules over an Algebra" has been deleted. The material in the new chapter has been extensively revised to bring it up to date and also to make it slightly more accessible. Further improvements to this chapter are planned.

3 Algebraic Geometry

3.1 Schemes

New Features:

- Given a scheme X, the intrinsic `JacobianSubrankScheme`(X) determines the subscheme of X defined by the vanishing of the minors of the Jacobian matrix of X having size the codimension of X in its ambient.
- The intrinsic `IsHomogeneous`(X, f) now applies when the scheme X is is defined over a 1-dimensional function K.
- The intrinsic `Multidegree`(X, f) now applies when the scheme X is is defined over a 1-dimensional function K.
- Given an affine or projective scheme X of dimension at least 1 that is defined over an algebraic function field K, and a place pl of K, the intrinsic `IsLocallySolvable`(X, pl) returns whether X contains a point defined over the completion of K at pl.
- For hypersurfaces X and Y lying in the same ambient space, `CommonComponent`(X, Y) returns the (possibly empty) maximal hypersurface lying in the intersection of X and Y.

New Features: Additional Access Functions

- The intrinsic `Degrees`(X, f) returns the sequence of homogeneous degrees of the polynomial f with respect to the gradings on the scheme X.
- The intrinsic `IdenticalAmbientSpace`(X, Y) returns `true` if the schemes X and Y lie in the same ambient space.
- The intrinsic `HasGCD`(X) returns `true` if GCDs can be computed for multivariate polynomials over the base ring of scheme X.
- The intrinsic `HasGroebnerBasis`(X) returns `true` if Gröbner bases can be computed for multivariate polynomial ideals over the base ring of scheme X.
- The intrinsic `HasResultant`(X) returns `true` if resultants can be computed for multivariate polynomials over the base ring of scheme X.

Changes:

- Improvements have been made to the error checking for `FunctionField`.

Bug Fixes:

- The intrinsic `IsomorphicProjectionToSubspace` has been fixed so that the projective ambient of the projected variety has a coordinate ring with the correct grevlex monomial ordering. The incorrect ordering in the old version caused some bugs and other problems.

3.2 Sheaves

New Features:

- The intrinsic `IneffectiveRiemannRochBasis`(X, I, J) is an extension of the `RiemannRochBasis` intrinsic that computes the sheaf or Riemann-Roch basis for a non-effective divisor that can be represented as the difference $D - E$ of two effective locally principal divisors D and E which have no irreducible component in common.

3.3 Algebraic Curves

New Features:

- `DeltaAdjustment` computes the amount that a singular point (possibly over a finite extension) on a curve adds to the arithmetic genus of the singular curve over the genus of its normalisation. Computed from the local Hilbert series of the singularity.
- `GenusViaArithmeticGenus` gives an alternative intrinsic to compute the genus of (the normalisation of) a projective curve by performing the much simpler computation of its arithmetic genus and subtracting off the delta adjustments of all of its singular points.

Bug Fixes:

- A crash when constructing places from a point on a curve has been fixed. (V2.24-2)
- A bug has been fixed in the Shanks group order routine, used in some cases to compute the order of the Jacobian of a hyperelliptic curve over a finite field. This could cause either crashes or wrong answers. (V2.24-2)
- The incorrect setting of the `GeometricallyIrreducible` attribute when setting the `IsNonsingular` attribute on an affine curve has been fixed. (V2.24-2)
- A problem with non-monic models in cyclic covers has been fixed. (V2.24-4)
- The `MonicModel` intrinsic now additionally returns a transformation. (V2.24-10)

3.4 Algebraic Surfaces

New Features:

- Intrinsics `IntersectionNumberOfStrictTransforms` and `SelfIntersectionOfStrictTransform` give the intersection number for the strict transforms of effective divisors on the blow-up desingularisation of any ordinary projective surface with only point singularities. This completes existing functionality that gives the intersection numbers of the irreducible blow-up divisors amongst themselves and with the strict transforms of effective divisors.
- `BasisOfHolomorphicTwoForms` computes a "base" meromorphic differential 2-form w and a basis for the everywhere-defined 2-forms in the form $f_1 w, \ldots, f_r w$, where f_i are rational functions, on the blow-up desingularisation of an ordinary projective surface with only point singularities. The divisor of w is also computed and returned.
- `PluriCanonicalBasis` computes a basis for everywhere defined n-tensor 2-forms, $n \geq 2$.

- `ExceptionalDivisors` returns all (-1)-curves (including irreducible curves that split into a conjugate set of (-1)-curves over a finite extension of the base field) on the desingularisation of an ordinary projective surface with only point singularities that is not rational or birationally-ruled. It also has a secondary return value containing all of the other irreducible curves that are contracted in the map from the desingularisation of the surface to its minimal model.
- For the blow-up resolution over an isolated singular point on a surface, `DualResolutionGraph` computes the graph with vertices corresponding to the irreducible blow-up divisors and edges corresponding to intersections between these and edge-loops to singular points on these as a Magmamultigraph. The labels on edges and vertices give additional information, like local intersection numbers and genus and arithmetic genus of the divisors. `MinimalDualResolutionGraph` gives the corresponding graph for the *minimal* resolution over the singular point. There are a number of related new intrinsics to conveniently access this additional information for edges/vertices of the graph.
- For the blow-up resolution over an isolated singular point on a surface over a finite field, intrinsics `NumberOfPointsOnResolutionFibre` and `ZetaFunctionOfResolutionFibre` compute the number of points or the (rational) zeta-function of the reduced divisor consisting of the union of all of the irreducible blow-up divisors on the desingularisation over the point. These are very useful for point-counting on the desingularised surface.
- A new package of functionality to work with degree 2 K3 surfaces in characteristic $\neq 2$. These are K3 surfaces presented as the desingularisation of a double cover of the projective plane ramified over a plane sextic. The basic constructor `DegreeTwoK3Surface` takes the equation of the sextic as an argument, checks that it has only simple (curve) singularities, and returns the (generally singular) model of the surface, naturally embedded in a $\mathbf{P}(1,1,1,3)$ weighted projective space, with the singular subscheme correctly filled in.
- `TriTangentLines` and `SixTangentConicsModp` are intrinsics used to find (-2)-curves on a degree 2 K3 surface S. The first returns all of the lines in the plane that split in S when pulled back under the degree 2 covering map. The second is a prototype version of the corresponding function for conics in the plane. It only works over (small) finite fields, using enumeration of possibilities, but often reveals split conics over $\mathbf{Q}$ when applied to the reduction at several small primes.
- `IntersectionMatrixOnDegree2K3` computes the full intersection matrix of the set of divisors consisting of the strict transforms of a given collection of irreducible curves on a degree 2 K3 surface along with all of the irreducible ((-2)-curve) blow-up divisors over singular points on the ramification sextic. This, together with the last two intrinsics listed, can be used for the partial (or complete!) computation of the Neron–Severi group of the surface.
- Given a finite collection $\mathcal{C}$ of (-2)-curves on a degree 2 K3 surface S, including the irreducible blow-up divisors if there are any, `KodairaConfigurations` computes "configuration" data for all distinct elliptic fibrations of S which contain at least one bad fibre consisting entirely of (-2)-curves lying in $\mathcal{C}$. This is a practical method of finding many elliptic fibrations if a reasonable number of (-2)-curves can be found using the above methods combined with the blow-up curves arising from desingularisation.
- `EllipticFibrationRRSpaceDeg2K3` explicitly computes the elliptic fibration map for any configuration returned by the previous intrinsic.
- Given a fibration map on a degree 2 K3 surface S as returned by the previous intrinsic, `EllipticGeneralFibreDeg2K3` computes a singular model of the genus 1 generic fibre of the map. This gives a genus 1 curve over a rational function field $k(t)$ for which the surface S is a global minimal model. If a sequence of sections of the fibration are also given, as subschemes of S, the intrinsic returns the sequence of $k(t)$-points on the generic fibre that correspond to the sections. Using a $k(t)$-point on the generic fibre as a base point, standard Magmaintrinsics can then be used to transform to a Weierstrass elliptic curve model.

New Features – Auxiliary Intrinsics:

- Given an ordinary projective plane $P2$ over a field and an indexed set S of distinct points on $P2$, the intrinsic `CollinearPointsOnPlane`$(P2, S)$ returns a sequence containing all subsets (as enumerated sequences) having three or more points of S that are collinear.
- Given an ordinary projective plane $P2$ over a field and an indexed set S of 8 or fewer distinct points on $P2$, the intrinsic `PointsInGeneralPosition`$(P2, S)$ returns whether the points in S are in general position.
- Given a degree 3 Del Pezzo surface X, the intrinsic `EckardtPoints`(X) computes the set of Eckardt points of X. These are points where three coplanar lines of the surface meet.

Changes:

- `KodairaDimension` and `KodairaEnriquesType` have been generalised from ordinary projective surfaces with only simple singulatities to ordinary projective surfaces with arbitrary point singularities in $\mathbf{P}^n$, $n \geq 4$ or arbitrary singularities for hypersurfaces in $\mathbf{P}^3$. The boolean `CheckADE` parameter has been changed to a boolean `KnownADE` parameter.
- `CanonicalIntersection` (for the strict transform of a surface divisor), `HomAdjoints`, `PlurigenusOfDesingularization`, `ArithmeticGenusOfDesingularization`, `IsRational` and `FirstChernClassOfDesingularization` have all been generalised from hypersurfaces in $\mathbf{P}^3$ to any ordinary projective surface S with the proviso that S only has point singularities if it lies in $\mathbf{P}^n$, for $n \geq 4$.
- `HasOnlySimpleSingularities` has been changed to avoid the use of algebraically-closed fields to split the singular subscheme. This caused errors for some base fields that should now no longer occur. Another change is that now the `ReturnList` option only returns one representative for each conjugate set of simple singular points over the base field.

Bug Fixes:

- A bug was fixed (fix originally exported in patch release V2.24-4) in `ResolveSingularSurface` for the blow-up desingularisation case, which occasionally created erroneous duplicate copies of irreducible blow-up divisors. A related bug in `IntersectionMatrix` for the blow-up divisors was also fixed. This didn't completely solve all duplicate divisor problems and a further fix has since been added for this release that should finally resolve the issue. Thanks to D. Lorenzini for providing bug examples.

4 Arithmetic Geometry

4.1 Binary and Ternary Forms

New Features:

- Tools for the minimization and reduction of binary forms defined over the rationals or integers have been provided by M. Stoll. The main intrinsic is `MinRedBinaryForm`(f), where f is a homogeneous polynomial in two variables defined over the rational field.
- A new package for the minimization and reduction of ternary forms over the rationals developed by S. Elsenhans and M. Stoll, has been added. The algorithm can be invoked via the intrinsic `MinimizeReduce`.

4.2 Rational Curves and Conics

Bug Fixes:

- An occasional problem with an uninitalized array with Lagrange's method over number fields has been fixed. (V2.24-9)

4.3 Elliptic Curves

4.3.1 Elliptic Curves over the Rational Field

New Features:

- The Cremona database of all elliptic curves of small conductor has been updated to include all conductors up to 499,999. In this update curves with conductors in the range 400000 to 500000 have been added thereby introducing an additional 581,056 curves in 423,257 isogeny classes. Important: ten isogeny classes have been slightly renumbered so that the first curve in the class is the optimal one. The affected classes are: 235470bb, 235746u, 258482a, 265706a, 333270bu, 359282a, 369194a, 375410g, 377034t, and 389774b. The database is accessed via the intrinsic `EllipticCurveDatabase`.

Bug Fixes:

- A p-adic precision problem with `IntegralQuarticPoints` in the case of high rank curves was resolved. (V2.24-3)
- The `FaltingsHeight` in the case of curves with non-integral j-invariant has been corrected. It now agrees with `FaltingsHeight2` though neither is normalized as per Deligne. Also, the Handbook gives an incorrect formula this. (V2.24-9)
- A problem with the interaction of the `ThreeDescent` machinery with recent changes to real lattices (`IsPositiveDefinite` and `LLLGram` now can give runtime errors) has been fixed.

4.3.2 Elliptic Curves over Number Fields

New Features:

- The `EllipticCurveSearch` intrinsic now has parameters to skip the 2-cover reduction and/or Cassels-Tate stage, and also to request more effort be put into searching on 2-covers. (V2.24-10)

Bug Fixes:

- A problem with LLL precision in `EllipticCurveSearch` has been remedied via the usage of try/catch. (V2.24-8)
- Some problems with infinite loops in `EllipticCurveSearch` (involving no suitable `ShortVectors`] ever being found) were fixed. (V2.24-10)
- A bug was fixed in the `EllipticCurveSearch` intrinsic. This could cause problems with curves over non-quadratic fields with rational j-invariant when these were related to an elliptic curve over $\mathbf{Q}$ with 2-conductor neither 4 nor 6. (V2.24-10)

4.3.3 Elliptic Curves over Finite Fields

Bug Fixes:

- A crash when computing the discrete logarithm (intrinsic `Log`) of a non-identity point with respect to the identity point has been corrected. (V2.24-3)

4.3.4 Elliptic Curves over Function Fields

Bug Fixes:

- A bug with computing the `MordellWeilGroup` when the order of the torsion group was not coprime to the extension degree of the field of the geometric rank was fixed. (V2.24-2)

4.4 Hyperelliptic Curves and Jacobians

New Features:

The following upgrade of the machinery for genus 2 and hyperelliptic curves has been provided by M. Stoll.

- The code for computing heights of points on genus 2 Jacobians over **Q** has been completely overhauled. The approach follows the paper J.S. Müller, M. Stoll: *Canonical heights on genus two Jacobians*, Algebra and Number Theory 10, No. 10, 2153-2234 (2016). In particular, the new factorization-free algorithm for computing canonical heights is used together with an improved and more efficient height constant. The main intrinsics are `HeightConstant`, `NaiveHeight` and `CanonicalHeight`.
- The intrinsic `ReducedBasis` has been completely rewritten using the new height code. Given a sequence of points on a hyperelliptic Jacobian, the intrinsic computes a LLL-reduced basis (with respect to the height pairing) of the subgroup generated by the points modulo torsion.
- The intrinsics `HeightPairing` and `HeightPairingMatrix` have been made more efficient in the case of Jacobians of genus 2 curves over **Q** and rational function fields by reducing the number of calls to `CanonicalHeight`.
- Functionality has been added for computing with *double Richelot isogenies* over the rationals, i.e., compositions of two Richelot isogenies such that the composition is defined over **Q**, but the individual Richelot isogenies are not. The main intrinsic is `DoubleRichelotIsogenies`.
- Also included is the intrinsic `TwoPowerIsogenies` which computes principally polarised abelian varieties of dimension two which are isogenous over the rationals to a genus 2 Jacobian over the rationals by an isogeny of degree a power of two.
- For the first time an intrinsic `MordellWeilGroupGenus2` has been provided which computes the Mordell-Weil group of a hyperelliptic Jacobian of genus 2 over the rationals.
- Mmore generally, the Mordell-Weil group of a hyperelliptic Jacobian of genus 1 or 2 over, respectively, a number field or the rationals, can be computed using the intrinsic `MordellWeilGroup`.

Bug Fixes:

- A problem with an reducible global lift in a p-adic point counting method has been resolved with `LPolynomial`. (V2.24-7)
- A bug was fixed in the Selmer machinery, which occasionally caused wrong results for `RankBounds` to be returned. A preconditioning attempt to use a better model in `MonicModel` was accidentally replacing the curve by a nonisomorphic one. (V2.24-9)
- A problem with heights on genus 2 curves, involving trying to compute the regular model (when it is not needed), has been fixed. (V2.24-10)

4.5 Hypergeometric Motives

New Features:

- An improved recognition method of the cyclotomic case for Jacobi motives has been applied, and an alternative method to reconstruct Euler factors from their p-adic roots is used in large degree. (V2.24-4)

5 Global Fields

5.1 Algebraic Number Fields

New Features:

- The parameter `Al` for intrinsic `MaximalOrder` now accepts the string `"Montes"` to indicate that the Montes algorithm is to be used to compute the maximal order of a number field. This parameter value for `Al` is also used for intrinsic `MaximalOrder` when applied to orders. The number field or order must be a simple extension of **Q** or **Z**, respectively.
- The parameter `Al` for intrinsic `Decomposition` now accepts the string `"Montes"` to indicate that the Montes algorithm is to be used to decompose a prime in a number field. This parameter value for `Al` is also used for intrinsic `Decomposition` when applied to orders. The number field or order must be a simple extension of **Q** or **Z**, respectively.
- A power product representation has been introduced for images under the map that is returned by `ClassGroup`. This representation is used if the parameter `UsePowerProduct` is set to `true` when invoking `ClassGroup`.
- The power product representation can be used in the computation of a `PicardGroup`. This reduces the CPU time when large powers of ideals arise during the computation.
- The boolean-valued intrinsic `IsKummerExtension` has been provided to determine whether a given field or order is a Kummer extension.
- The algorithm to compute the inverse of an element in a relative extension has been greatly improved in the high degree case.

Changes and Removals:

- The algorithm selection for `Factorization` of polynomials over number fields, orders of number fields and fields of fractions of orders of number fields has been reviewed. Algorithm selection is now based on whether the basis for the field or order, represented as a direct extension of **Q** or **Z**, consists of powers of one element. (Partially in V2.24-5)

Bug Fixes:

- A crash in `IdealsUpTo` has been fixed. (V2.24-3)
- Testing the zero ideal for being square is now handled. (V2.24-4)
- Computing GCDs of elements of quotients of an order of a number field by an ideal has been fixed. (V2.24-4)
- A crash in `pFundamentalUnits` has been fixed. (V2.24-4)
- A crash with the number field sieve (quadratic case) has been fixed, involving multiple divisors of the discriminant in the range around 1000 causing a buffer overflow. (V2.24-9)

5.2 Characters and Artin Representations

Bug Fixes:

- A problem with failing to find distinct ST polynomials has been fixed. (V2.24-10)

5.3 Algebraic Function Fields

New Features:

- The parameter `Al` for intrinsic `MaximalOrderFinite` now accepts the string `"Montes"` to indicate that the Montes algorithm is to be used to compute the finite maximal order of an algebraic function field. This value for parameter `Al` is also used for intrinsic `MaximalOrder` when applied to orders. The function field or order must be a simple extension of a rational function field or a polynomial ring, respectively.
- The parameter `Al` for intrinsic `Decomposition` now accepts the string `"Montes"` to indicate that the Montes algorithm is to be used to compute the decomposition of a place in an algebraic function field. This value for parameter `Al` is also used for intrinsic `Decomposition` when applied to decompose a prime polynomial in an order. The function field or order must be a simple extension of a rational function field or a polynomial ring, respectively.
- An `Al` parameter has added to intrinsic `Genus` which if set to `"Montes"` will use a Montes algorithm for the computation.
- A parameter `SeparatingElement` has been added to intrinsic `FunctionField` to allow for the input of a polynomial (type `RngMPolElt`). This allows the user to provide an indeterminant as the separating element of the resulting extension.
- The boolean intrinsic `IsKummerExtension` has been provided to determine whether a given field or order is a Kummer extension.

Bug Fixes:

- Testing the zero ideal for being square is now handled. (V2.24-4)

5.4 Galois Groups

New Features:

- The intrinsic `GaloisSplittingField` has been introduced to compute a splitting field for polynomials over global rational function fields.
- The intrinsic `SolveByRadicals` has been extended so as to apply to polynomials over global rational function fields.

Changes:

- Improvements have been made to the error checking for the `GaloisProof` intrinsic.

Bug Fixes:

- A problem with large degree number fields with a medium size subfield has been fixed. For example, degree 96 fields with a subfield of degree 16 can now be handled.
- Tschirnhaus transformations are now allowed to use higher degree polynomials during the computation of `GaloisSubgroup`. (V2.24-10)

6 Local Fields

Changes:

- Factorization of polynomials over infinite precision rings will increase default precision to the minimum precision of the coefficients of the polynomial in an attempt to avoid errors arising due to a lack of precision.
- The computation of `Log` has been made more accurate.

6.1 Series Rings

New Features:

- It is now possible to provide a coefficient ring homomorphism when constructing homomorphisms from extensions of series rings. (V2.24-9)

7 Basic Rings and Fields

7.1 Integer Ring

New Features:

- In 64-bit mode, a large integer may now have up to 2^{69} bits (the previous limit was up to 2^{37} bits). This also allows multiplication of higher degree univariate polynomials (over the integers and integer residue rings, etc.) than was previously possible.
- Some memory is now saved for some low-level arithmetic operations when working with integers having sizes between 64 and 128 bits.
- A modification to the factorization routine has been applied to remove large powers more efficiently. (V2.24-4)
- The universe of the sequence returned by Partitions has been changed to be consistent with other sequences of sequences. (V2.24-6)
- The function `Random(a, b)` has been significantly sped up. This also leads to speedups when constructing large random objects with small entries (for example, a large matrix over a small finite field).

Bug Fixes:

- A slowdown in `Modexp` for integers when the modulus had a certain form has been fixed.

7.2 Real and Complex Fields

New Features:

- Real and Complex fields may now be included in sets.

Bug Fixes:

- The deprecated `LinearRelation` intrinsic now accepts sequences and vectors over the reals. (V2.24-2)
- A bug in root finding for a non-monic binomial polynomial has been fixed. (V2.24-8)

7.3 Polynomial Rings

New Features:

- The general factorisation algorithm for univariate polynomials over $\mathbf{Z}$ or $\mathbf{Q}$ has been greatly sped up for several types of input, based on the following:
  1. Detection of the case that scaling of the main variable reduces the coefficients of the input polynomial.
  2. A heuristic to find low-cardinality combinations with less initial Hensel lifting.
  3. Faster construction of the polynomials arising from the combinations found by the van Hoeij algorithm (particularly faster for higher degree polynomials).
  4. Faster construction of the final factors over the number field.
- The Trager factorisation algorithm for univariate polynomials over an algebraic number field $K = \mathbf{Q}(\alpha)$ has been greatly sped up for several types of input, based on the following:
  1. Better choice of scalar shift of the main variable to find a squarefree norm.
  2. Faster construction of evaluation of polynomials at scalar shifts of the main variable.
  3. Faster construction of the integral norm polynomial for certain types of number field.
- The Brent-Kung algorithm for modular evaluation has ben been significantly sped up for polynomials defined over large prime finite fields. As a result, the algorithm for factoring polynomials over large prime finite fields has also been significantly sped up.
- The modular GCD algorithm for univariate polynomials over an algebraic number field $K = \mathbf{Q}(\alpha)$ greatly sped up for several types of input.
- The squarefree factorisation algorithm has been improved for larger characteristic p fields. This can also speed up the initial squarefree phase of the general factorisation algorithm over such fields.

Changes:

- The function Degree has been fixed to return -1 when a grading is present and the input polynomial is zero.

Bug Fixes:

– Some crashes involving elements of quotients of multivariate polynomial rings have been fixed.

– An obscure bug in polynomial multiplication of high degree polynomials over `GF(2)` has been fixed.

– An crash in a rare situation for polynomial multiplication over the integers or an algebraic number field has been fixed.

– A hang when factoring a certain class of non-monic polynomial over algebraic number fields has been fixed. (V2.24-5)

8 Coding Theory

8.1 General Linear Codes

Changes and Removals:

– Automorphism group/isomorphism testing for binary codes is now treated as a 0/1 matrix automorphism group/isomorphism testing problem which is solved using B McKay's `nauty` program.

Bug Fixes:

– A bug in automorphism group/isomorphism testing computations for binary codes has been fixed. In the case of some very large codes it could sometime give a wrong answer. A new implementation of a backtrack search algorithm in the style of Jeff Leon is now being used.

– A crash in the intrinsic `IsPure` for quantum codes has been fixed. (V2.24-2).

9 Commutative Algebra

9.1 Local Polynomial Rings

New Features:

– For local polynomial rings, the function `Coordinates(I, f)` returns C and d so that `&+[C[i]*B[i]: i in [1 .. #B]] eq d*f` where `B := Basis(I)`.

Bug Fixes:

– Bugs in local polynomial rings with one variable have been fixed.

9.2 Gröbner Bases

New Features:

– The dense F_4 Gröbner basis algorithm now has multithreading support (based on multithreading support for the underlying linear algebra) for dense ideals defined over the field GF(p) for prime $p \leq \lfloor 2^{23.5} \rfloor = 11863283$. To specify that k threads are to be used, the following statement should first be given:

```
SetNthreads(k);
```

Bug Fixes:

– A blowup in time/memory for Groebner basis computation in the case of sparse zero-dimensional ideals of high degree has been fixed. (V2.24-6)

9.3 Ideal Theory

New Features:

– The algorithm to compute the radical of a positive-dimensional ideal has been greatly sped up for certain classes of inputs.

10 Complex Manifolds

10.1 Riemann Surfaces

A package developed by Christian Neurohr for computing with Riemann surfaces defined by an affine equation $f(x, y) = 0$ is included in this release. The algorithms are described in his 2018 PhD thesis at Universität Oldenburg:

`https://oatd.org/oatd/record?record=oai%5C%3Aoops.uni-oldenburg.de%5C%3A3607`

Features:

– The package introduces a Riemann surface type `RieSrf` together with types for points `RieSrfPt` and divisors `DivRieSrfElt`.

– An object of Riemann surface type is defined in one of two ways:

  – As a general curve defined by an irreducible polynomial $f \in K[x, y]$ and a mapping P where K is $\mathbf{Q}$ or a number field, and P is a complex embedding;

  – As a superelliptic curve defined by $f = y^m - h(x)$ with $h \in C[x]$ squarefree and $m > 1$.

– Working with a precision of several hundred decimal digits is practical in the general case and several thousand digits in the superelliptic case.

– Period matrices and the Abel-Jacobi map of superelliptic curves will be much faster than the general ones.

– Several different algorithms for numerical integration are available, namely: Double-exponential, Gauss-Legendre or Clenshaw-Curtis quadrature in the general case and double-exponential or Gauss-Jacobi quadrature in the superelliptic case.

– The most important intrinsics for Riemann surfaces are: `RiemannSurface`, `SmallPeriodMatrix`, `BigPeriodMatrix`, `FundamentalGroup`, `AnalyticContinuation`, `MonodromyRepresentation`, `HomologyBasis`, and `AbelJacobi`.

11 Group Theory

11.1 Finitely Presented Groups

New Features:

- The intrinsic `AutomorphismGroup` now applies to free groups. In this case it returns the automorphism group as a group of mappings A, the automorphism group as a finitely-presented group G, and the isomorphism from A to G.
- The test for an fp-group of being hyperbolic (`IsHyperbolic`) has been expanded to compute and apply a Dehn algorithm. The Dehn algorithm is returned via three additional return values.
- The intrinsic `SimpleQuotients` which searches for simple group quotients of a finitely presented group has been upgraded to search for simple quotients of order up to 10^{10} (up from 10^9).
- The intrinsic `LowIndexNormalSubgroups`, which searches for normal subgroups of small index in a finitely presented group, has been upgraded from an index limit of 50,000 to 100,000. Note that the intrinsic can be applied with larger index limits in which case there is no guarantee that all normal subgroups of index up to that limit will be found.

11.2 Finite Groups

New Features:

- Given a group G and a subgroup H of G, the intrinsic `SubgroupFusion`(G, H) returns the fusion of the conjugacy classes of H in the conjugacy classes of G.
- Given two groups G and H, the intrinsic `IsIsomorphicToSubgroup`(G, H) returns `true` if H is isomorphic to a subgroup of G.

11.3 Permutation Groups

New Features:

- A canonical form for permutation groups is now available.
- The database of transitive groups has been extended to include the 195 826 352 transitive groups of degree 48 which were recently constructed by Derek Holt. So the transitive group database now includes the transitive groups for all degrees less than 49. Some relevant intrinsics are `TransitiveGroups`(d) and `TransitiveGroups`(d, n), where d is the degree of the transitive groups sought and n is the number of a transitive group of degree d. It should be noted that, because of its size, the database of transitive groups of degree 48 is not included among the standard databases included in the download file `shared_complete.tar.gz` and so has to be downloaded separately from the Magma download site.

11.4 Matrix Groups

New Features:

- A new version of the Composition Tree package developed by Eamonn O'Brien and others has been installed. The major new feature is the inclusion of tools to recognize exceptional groups.
- A package developed by Bettina Eick, Tommy Hofmann and Eamonn O'Brien for testing conjugacy of elements in $GL(n, \mathbf{Z})$ has been developed. The centraliser of an element of $GL(n, \mathbf{Z})$ can also be computed. The relevant intrinsics are `AreGLConjugate`(A, B) and `CentralizerGLZ`(A).
- The machinery for computing complements of a normal subgroup N in a permutation group G has been extended to complements of a normal subgroup in a finite matrix group. The relevant intrinsic is `Complements`(G, N).
- An efficient algorithm for computing the order of a matrix having entries in the ring $\mathbf{Z}/n\mathbf{Z}$, for any positive integer n, has been implemented. (Previously a naive brute force search algorithm was used.)

11.5 Classical Groups

New Features:

- Maximal subgroups have been installed for a the following seven classical groups and their almost simple extensions: $L_8(3)$, $L_9(3)$, $U_5(4)$, $U_6(3)$, $S_{14}(2)$, $O^+_{14}(2)$, $O^-_{14}(2)$. These were contributed by Derek Holt.
- The automorphism groups for the seven classical groups above have been installed.
- The short presentations of Leedham-Green and O'Brien for classical groups on their standard generators are now available directly, and are used in the verification of composition trees for matrix groups.

11.6 Exceptional Groups

New Features:

- Constructive recognition for those families of exceptional groups not previously implemented and having rank at least 2 is now available. The intrinsic is `ExceptionalConstructiveRecognition`.
- Standard generators for these exceptional groups are constructed using algorithms of Liebeck and O'Brien. The corresponding intrinsic is `ExceptionalStandardGenerators`.
- Elements of these exceptional groups are written as words in their standard generators using algorithms of Cohen, Murray and Taylor. The corresponding intrinsic is `ExceptionalRewrite`.
- Presentations for these exceptional groups on their standard generators are also available. The corresponding intrinsic is `ExceptionalStandardPresentation`.
- Equivalent machinery for the remaining families of exceptional groups was already available. The above tools for exceptional groups were implemented by Eamonn O'Brien and Don Taylor.
- Maximal subgroups and automorphism groups have been installed for the two exceptional groups $F_4(2)$ and Ree(27) by Derek Holt.

11.7 Sporadic Simple Groups

New Features:

- Maximal subgroups and automorphism groups have been installed for the sporadic groups HN and Fi_{23} by Derek Holt.

12 Lattices and Quadratic Forms

12.1 Lattices

New Features:

- Vector enumeration in parallel (pthreads) can now be done with `UseParallelVectorCode`.

Changes and Removals:

- Various lattice functions involving integral lattices (such as enumeration) now use information about the evenness of a lattice when applicable. (V2.24-2)

Bug Fixes:

- A problem with transforms in `BKZ` has been fixed. (V2.24-3)
- The `BKZ` intrinsic now gives an error when stability problems are detected, rather than simply returning the partially reduced lattice. (V2.24-4)
- A possible usage of uninitialized memory in `BKZ` was removed. (V2.24-6)
- A problem with `Sphere` (over number fields) was fixed. (V2.24-9)

12.2 Binary Quadratic Forms

Bug Fixes:

- The `ClassGroup` computation for quadratic forms with discriminant 1 mod 4 has been corrected. (V2.24-8)

13 Lie Theory

13.1 Curtis–Steinberg–Tits Presentations

New Features:

- Given a type, rank, a field size q and a q-restricted weight λ, the intrinsic `CST_Generators` returns matrix generators in a highest weight representation of weight λ for the corresponding finite group of Lie type. The generators are in Curtis–Steinberg–Tits (CST) format.

– The intrinsic `CST_Presentation` returns the CST relations for a finite group of Lie type as a sequence of straight line programs.

– `CST_VerifyPresentation` checks whether a given collection of matrices in CST format satisfies the CST relations for the group.

– Given CST generators which satisfy the presentation for a group of Lie type the intrinsic `CSTtoChev` returns a map from the group they generate to the standard Magma copy obtained from the intrinsic `ChevalleyGroup`. The map is projective; i.e., a homomorphism up to scalar multiples.

Changes and Removals:

– The intrinsic `TwistedGroupOfLieType` now accepts a type, a rank and the size of the defining field and returns the twisted group of Lie type with a simply connected root datum.

– The intrinsic `Random` has been extended to return random elements of finite twisted groups of Lie type.

– `PapiOrder` is a variant of `AdditiveOrder`.

13.2 Lie Algebras

Changes and Removals:

– Added Handbook entries for the following intrinsics which have been present in the Magma package files for several releases but undocumented:

  – `DominantWeights` (previously called `DominantCharacter`),

  – `WeylDimension` (previously called `DimensionOfHighestWeightModule`),

  – `DecomposeTensorProduct`,

  – `DecomposeSymmetricPower`,

  – `DecomposeExteriorPower`,

13.3 Representation Theory

New Features:

– Given a representation of a finite group of Lie type, `ContravariantForm` returns a symmetric contravariant form on the representation space. `ContravariantFormSpace` returns the space of all contravariant forms.

– `HighestWeight` returns the highest weight and a maximal vector of a highest weight module of a group of Lie type.

– Given a representation of a group of Lie type, `UnipotentFixedSpace` computes the space of fixed points of the unipotent radical of a standard Borel subgroup.

– `FrameBase` returns weights $\lambda = \lambda_1, \lambda_2, \ldots, \lambda_k$, in the Weyl group orbit of a weight λ, group elements $w_1 = 1, w_2, \ldots, w_k$ such that $\lambda w_i = \lambda_i$ and subsets $I = J_0 \supset J_1 \supset \cdots \supset J_k = \emptyset$ of simple reflections such that $W_{\lambda_1,\ldots,\lambda_i} = W_{J_i}$ for $1 \leq i \leq k$.

– Given matrix generators for a group of Lie type, one for each simple root, `ExtendGeneratorList` returns generators in CST format.

- Given a representation ρ of an untwisted simply connected finite group G of Lie type and a matrix A in the image, the intrinsic `ChevalleyForm` returns the components of the Chevalley normal form of $g \in G$ such that $\rho(g) = A$. The intrinsic `TwistedChevalleyForm` does the same for twisted groups.
- `Morphism` constructs a homomorphism from a group of Lie type (twisted or untwisted) to a matrix group defined by generators in CST format.
- For a homomorphism ρ from an untwisted finite group of Lie type to a matrix (such as that returned by `Morphism`), the intrinsic `RowReductionMap` returns a function f from the matrix group to the group of Lie type such that $\rho(f(A)) = A$ for all matrices A. `TwistedRowReductionMap` provides the same functionality for twisted groups.
- For a finite simply connected group of Lie type over a field of size q and a q-restricted weight, `IrreducibleHighestWeightRepresentation` returns the irreducible highest weight representation of weight λ.

14 Linear Algebra and Module Theory

14.1 Linear Algebra Over Finite Fields

New Features:

- Multithreading is now supported for multiplication of large matrices over the field GF(p) for prime $p \leq \lfloor 2^{23.5} \rfloor = 11863283$. To specify that k threads are to be used, the following statement should first be given:

```
SetNthreads(k);
```

- The base algorithm for matrix multiplication in large dimension has been significantly improved for matrices over GF(2^k) for $k = 2, 3, 4$.
- A major improvement in both time and memory usage has been achieved for multiplication of large matrices over GF(q) where $q = p^k$, $p > 2, k > 1$, and $q < 256$.
- Matrix transpose over GF(q) for $q = 3, 4, 5, 7, 8$ has been significantly improved (for all matrix sizes).
- Matrix multiplication over GF(p) for $p = 3, 5, 7$ has been significantly improved (for all matrix sizes) in the AVX version.

Bug Fixes:

- The CUDA algorithm for matrix mulitplication over GF(p) for $p = 3, 5, 7$ has had a fix in very large dimension.

14.2 Linear Algebra Over General Rings

New Features:

- The recursive echelon algorithm has been improved: (1) in the case where a transformation matrix is required (particularly when the number of rows is less than the number of columns); and (2) in the case where the base ring is GF(q) for $q \leq 7$. This also yields improvements to the many algorithms which depend on this, including the computation of the inverse of a matrix.
- The modular algorithm for multiplication of matrices with large integer entries (including integers in large residue class ring) has been significantly sped up, particularly in the case that the size of the integers is large.
- The algorithm for multiplication of small matrices with rational entries has been sped up.
- The algorithm for echelonisation of matrices with rational entries has been sped up when the matrices are sparser.
- The nullspace algorithm for matrices with integer or rational entries has been significantly sped up in the case that the nullity is large.
- The computation of CharacteristicPolynomial over reals/complexes now goes via NumericalEigenvalues.

Bug Fixes:

- A check has been added to TensorProduct to detect when the resulting matrix would not fit into the available memory.

14.3 Sparse Matrices

New Features:

- The main sparse Gaussian elimination algorithm has been greatly improved for very large sparse matrices, in reduction of both memory usage and time taken. For sparse matrices with at least one dimension over a million, the improvement can be quite dramatic.
- Even greater reduction of memory and time usage in the main sparse Gaussian elimination algorithm has been achieved for very large sparse matrices over GF(2).
- New parameter `MakeWider` has been added to the function `Rank` for sparse matrices, to specify that the elimination algorithm should transpose the input if necessary to ensure that the number of columns is greater than or equal to the number of rows (this help a lot with certain types of input).

Bug Fixes:

- Some memory leaks in sparse matrix algorithms have been fixed.
- A bug in `IsDiagonal` has been fixed.

14.4 Modules Hom(U, V)

Bug Fixes:

- The function `Cokernel` was incorrectly returning only one result in assignment context; this has been fixed.
- A bug in `IsSurjective` has been fixed.

15 Linear Associative Algebras

15.1 Quaternion Algebras

Bug Fixes:

– A incorrect third return value with `IsConjugate` for orders was fixed. (V2.24-7)

15.2 Group Algebras

Bug Fixes:

– A problem with the map returned by VectorSpace for subalgebras of group algebras has been fixed. (V2.24-9)

16 Modular Forms

16.1 Classical Modular Forms

New Features:

– Major improvements have been made for computing newforms of cuspidal subspaces. For example, compared with V2.24, the statement `F := Newforms(CuspForms(2275, 4));` is now about 50 times faster.

– Even greater improvements have been made for computing newforms of cuspidal subspaces twisted by a Dirichlet character. For example, compared with V2.24, the statement

```
F := Newforms(CuspForms(G.1^2, 2)) where G := FullDirichletGroup(N);
```

is now about 670 times faster for $N = 139$, and about 1960 times faster for $N = 163$.

– The setting up of an absolute number field and the application of the appropriate isomophism (when needed) to compute q-expansions has been greatly improved. For example, the printing of the newforms which are returned by `Newforms(CuspForms(FullDirichletGroup(81).1^2, 2))` (and which involves the computation of q-expansions) is now about 110 times faster than for V2.24.

Bug Fixes:

– An erroneous caching with the usage of `EisensteinSeries` for forms with character has been fixed. (V2.24-4)

– A problem with `Basis` for half-integral weight forms has been fixed, involving auxiliary calculations in trivial spaces. (V2.24-4)

– A crash in NewformDecomposition involving very large number fields has been fixed. (V2.24-2)

– A crash with `NewformsOfDegree1` has been fixed. (V2.24-10)

– A problem with `NewformDecomposition` in square level has been patched. (V2.24-4)

– A numerical round-off problem with `InnerTwists` has been fixed. (V2.24-4)

16.2 Fuchsian Groups

Bug Fixes:

- A problem with parameters in `FuchsianGroup` has been fixed. (V2.24-9)

17 Numerical Analysis

17.1 Fourier Transform

New Features:

- Ihe discrete Fourier transform has been implemented for a sequence of complex numbers. The name of the relevant intrinsic is `DiscreteFourierTransform`.

17.2 Linear Algebra

New Features:

- A stable high-precision algorithm for computing the characteristic polynomial of a matrix over the real or complex fields has been implemented. The intrinsic name is `NumericalEigenvalues`. (V2.24-6)

17.3 Numerical Integration

New Features:

- High quality numerical integration code has been developed by Christian Neurohr. In particular, Gaussian, Clenshaw-Curtis and tanh-sinh integration methods have been implemented. the names of the intrinsics are `GaussLegendreIntegrationPoints`, `ClenshawCurtisIntegrationPoints` and `TanhSinhIntegrationPoints`.

18 Representation Theory

18.1 $K[G]$-Modules

New Features:

The following new intrinsics for $K[G]$-modules, where K is a finite field, have been added by Derek Holt.

- Intrinsic `HomMod`$(M,\ N)$: Given $K[G]$-modules M and N return $Hom(M, N)$ as a $K[G]$-module.
- Intrinsic `Inflation`$(M,\ \rho)$: Given a $K[H]$-module M and a group homomorphism $\rho : G \to H$, return M as a $K[G]$-module by inflation using ρ.
- Intrinsic FixMod(M, H): Given $K[G]$-module M and a subgroup H of G, return $\mathrm{Fix}(M_H)$ as $N_G(H)$-module.
- Also see new intrinsics `FixDual`(M), `FixDualMod`$(M,\ H)$.
- Intrinsic `Kernel`(M): Given the $K[G]$-module M, return the kernel of the corresponding representation.
- Intrinsic `MaximalExtension`$(M,\ N)$: Given $K[G]$-modules M and N, return the largest nonsplitting module extension of copies of N by M.
- Intrinsic `GTensorProduct`$(M,\ N)$: Given $K[G]$-modules M and N, return the tensor product $M \otimes_{K[G]} N$.
- Intrinsic `IsProjective`(M): Return `true` if the $K[G]$-module M is projective.
- Intrinsic `IsFree`(M): Return `true` if the $K[G]$-module M is free.
- Intrinsic `IsSelfDual`(M): Given the $K[G]$-module M, return `true` if M is self-dual.

Bug Fixes:

- A missing test for absolutely irreducibility has been added to IsRealisableOverSmallerField. (V2.24-7)
- A bug in the map returned by `AbsoluteRepresentation` has been fixed. (V2.24-8)

18.2 Brauer Characters

New Features:

A number of intrinsics for constructing and working with Brauer characters have been implemented.

- The intrinsic `BrauerCharacterTable` returns the table of p-modular Brauer characters.
- The intrinsic `BrauerCharacter` uses a very efficient algorithm for computing the Brauer character of a $K[G]$-module.
- Standard character arithmetic is supported.
- The transfer of Brauer characters between group-subgroup and group-quotient group are supported by intrinsics `Induction`, `Restriction` and `LiftCharacter`.
- The intrinsic `Blocks` partitions the table of ordinary characters into p-blocks. The defect group for a p-block can be constructed using the intrinsic `DefectGroup`.

18.3 $(K[G], K[H])$-bimodules

New Features:

$(K[G],\ K[H])$-bimodules have been implemented by taking advantage of the fact that in many respects they are equivalent to standard right $K[G \times H]$-modules.

- A type `LRModGrp` has been introduced for bimodules.
- Given $K[G]$- and $K[H]$-modules M and N of the same dimension with commuting actions the intrinsic `Bimodule`(M, N) constructs the corresponding $(K[G],\ K[H])$-bimodule.
- Intrinsics `LeftOppositeModule`(B) and `RightModule`(B) construct the left and right modules from the $(K[G],\ K[H])$-bimodule B.
- Most of the functionality for standard right $K[G]$-modules, such as forming sub- and quotient modules, defining homomorphisms, etc., also works for $(K[G],\ K[H])$-modules, where the results are calculated using the equivalent $K[G \times H]$-module.
- As tensor products on $K[G \times H]$-modules do not work in this approach to computing with bimodules, a bimodule tensor product intrinsic `TensorProduct`$(B1,\ B2)$ is provided.