Summary of New Features in Magma V2.8

July 31, 2001

1 Introduction

This document provides a terse summary of the new features installed in Magma for release version V2.8 (July 31, 2001). Previous releases of Magma were: V2.7 (June 30, 2000), V2.6 (November 8, 1999), V2.5 (July 7, 1999), V2.4 (December 3, 1998), V2.3 (January 30, 1998), V2.20 (April 18, 1997), V2.10 (October 14, 1996), V2.01 (June 21, 1996), and V1.30 (March 5, 1996).

2 Summary

Documentation

- *Handbook:* There are 18 new chapters in the Handbook.
- *Handbook:* The Handbook now includes bibliographies for each chapter, and some information concerning the algorithms used has been provided for a selection of the functions.
- *Help System:* A new help system has been developed by Claus Fieker. The system is html-based and allows the user to quickly access any Handbook entry via the usual help request from within Magma. To access the new help system, the following line should be placed in the user's startup file:

```
SetHelpUseExternalBrowser(true);
```

Groups

- *Permutation Groups:* An algorithm for determining the maximal subgroups of a permutation group is provided. This is applicable to any group for which the non-abelian composition factors either have order less than 1.6×10^7 or have a permutation representation of degree less than 32.
- *Permutation Groups:* The machinery for finding all conjugacy classes of subgroups has been extended to a much larger class of groups. In particular, the former limitation that the trivial Fitting quotient have order less than $216,000$ has been removed. Variations are provided to find all conjugacy classes of subgroups satisfying some condition.

- *Permutation Groups:* The maximal subgroup algorithm is used as the basis for an algorithm for determining all subgroups whose index is less than some given bound. This is much more efficient than finding all subgroups and selecting those which satisfy the index condition.

- *Permutation Groups:* A new algorithm for finding the automorphism group of a permutation group is also included. A variation may be used to determine whether two permutation groups are isomorphic.

- *Matrix Groups:* New algorithms for computing the conjugacy classes of elements and the maximal normal p-subgroup in a finite matrix group have been installed.

- *Matrix Groups:* Several functions for analysing the action of a matrix group over a finite field on subspaces of the underlying natural vector space have been introduced. These functions are designed to handle situations where the orbits may be large.

- *Finitely Presented Groups:* Much of the basic infrastructure for finitely presented groups has been redesigned, yielding improved performance and greatly increased applicability of key functions for fp-groups.

- *Finitely Presented Groups:* New versions of the algorithms for coset enumeration, simplification of presentations by Tietze transformations, Reidemeister–Schreier rewriting and computation of p-quotients have been installed. An interactive coset enumeration facility is now available.

- *Finitely Presented Groups:* The function `Order` for finitely presented groups has been revised and now uses a much more sophisticated strategy for determining the order of a (finite) group or for proving that the group is infinite.

- *Polycyclic Groups:* The functionality of the new category of polycyclic groups introduced in V2.7 has been greatly increased. In particular, algorithms have been implemented that compute the following: a normal series where the factors are either elementary abelian p-groups or free abelian groups; the Fitting subgroup; the Fitting series, the upper central series; the centre. For the case of nilpotent groups, functions that compute normalizers and centralizers have been installed.

- *p-Groups:* New implementations by Eamonn O'Brien of an algorithm for computing the automorphism group of a p-group and also an algorithm for generating p-groups have been installed. The new p-group generation implementation overcomes problems associated with file handling that affected the previous implementation.

- *Generic Abelian Groups:* A new category has been created for generic (finite) abelian groups. The creation of a generic abelian group is a new feature which allows the user to create an abelian group over any domain given that an identity and a group operation have been defined. Features include finding the order of an element, determining a presentation from generators, and computing the discrete logarithm of an element.

- *Subgroups of $PSL(2,R)$:* A package for working with congruence subgroups of $PSL(2,R)$ has been developed by Helena Verrill. The package includes graphics facilities which allow the user to produce pictures of fundamental domains.
- *Databases:* Several new databases of groups have been created, e.g. for perfect groups, almost simple groups, rational maximal finite matrix groups and finite quaternionic matrix groups.

Groups of Lie Type

- *Root Datum:* A data type has been introduced for root datum of groups of Lie type.
- *Coxeter Groups:* The module for Coxeter groups has been considerably expanded.
- *Lie Groups:* A data type has been introduced for groups of Lie type. The basic group operations have been implemented over any field.

Basic Rings

- *Integers:* An experimental implementation of the number field sieve (NFS) for factoring large integers is now available.
- *Integers:* The *Factor* database maintained by Richard Brent has been updated. It now contains $221,188$ factors f of integers $a^n \pm 1$, where $a < 10000, n < 10000$, and $f > 10^9$.
- *Univariate Polynomial Rings:* Polynomials over the integers or rationals are now factored by the new algorithm of Mark van Hoeij. The search for the correct combination of modular factors (which has exponential worst-case complexity in the standard Berlekamp–Zassenhaus algorithm) is performed by van Hoeij's algorithm, which efficiently finds the correct combinations by solving a Knapsack problem via the LLL lattice-basis reduction algorithm.

Extensions of Rings

- *Galois Rings:* Basic facilities for computing with Galois rings have been implemented.
- *Number Fields:* Algebraic number fields and their orders have been substantially revised in Magma V2.8. A new field type has been created for the field of fractions of an order of a number field. More functionality has been provided for both absolute and relative fields and orders.

- *Number Fields:* Factorization of univariate polynomials defined over number fields has become enormously more efficient through use of the van Hoeij ideas. It is now possible to factor a difficult degree 15 polynomial over a degree 15 extension of **Q** in a few seconds.

- *Number Fields:* Multi-step relative extensions are now fully supported with most operations for absolute extensions generalized for this situation. In particular, factorization and integral closure are now supported for relative extensions.

- *Number Fields:* Some restrictions on the creation of number fields (orders) have been removed. Thus, it is now possible to create a field or order using a non-monic polynomial. Further, it is possible to construct non-simple and linear extensions.

- *Number Fields:* Highly efficient arithmetic has been implemented for residue class rings. In addition, the calculation of unit groups of such rings is supported at least for quotients of absolute maximal orders.

- *Number Fields:* It is now possible to form the completion of a number field at a (finite) prime.

- *Number Fields:* Code implemented by Claus Fieker computes the p-Selmer group at a list of primes.

- *Number Fields:* Initial machinery has been provided for class field theory. In particular, a new type for *abelian fields* has been provided. It is now possible to construct ray class groups and class fields based on them (i.e. in terms of defining equations).

- *Number Fields:* It is now possible to compute the relative Galois group and relative subfields of a one-step relative extension of **Q**.

- *Quadratic Fields:* Quadratic fields have been redesigned so that they are now number fields with the addition of some special code implementing fast algorithms peculiar to quadratic fields. Thus, for the first time all number field operations are supported for quadratic fields.

- *Cyclotomic Fields:* Cyclotomic fields have been redesigned so that they are now number fields with the addition of some special code implementing fast algorithms peculiar to cyclotomic fields. Thus, for the first time all number field operations are supported for cyclotomic fields.

- *Function Fields:* Machinery for constructing and working with differentials has been introduced. It is possible to compute the space of holomorphic differentials for a given divisor. Other operations include computing the valuation of a differential at a place, computing the residue of a differential at a place of degree 1 and working with higher differentiations.

- *Function Fields:* A function that determines the sequence of Weierstrass places has been developed. Related functions compute the sequence of global gap numbers of a divisor or the sequence of gap numbers of a divisor at a place of degree 1.
- *Function Fields:* The divisor class group of a global function field may be computed using an algorithm and implementation due to Florian Heß. This also allows other related information to be determined: e.g., unit group, S-class group, S-units and S-regulator for a finite set of places S.
- *Function Fields:* Given the elliptic function field $E : y^2+xy+x^3+ax^2+b$ defined over $GF(q^n)$, a function is provided that computes a hyperelliptic function field in the Weil restriction of E over $GF(q)$. This feature is of particular interest to cryptographers.
- *Function Fields:* A generic algorithm is provided for the factorization of univariate and multivariate polynomials over function fields.
- *Function Fields:* The Galois group and lattice of subfields may be found for a function field using code implemented by Katharina Geißler.

Commutative Algebra

- *Ideal Theory:* It is now possible to compute Gröbner bases of ideals of polynomial rings defined over Euclidean rings (including the important case of the ring $\mathbf{Z}$). Such Gröbner bases are constructed by an extension of Jean-Charles Faugère's F_4 algorithm which uses sparse linear algebra (algorithm due to Allan Steel). Many of the standard functions provided for ideals in polynomial rings defined over fields are now generalized to ideals in polynomials rings defined over Euclidean rings.

Linear Algebra and Module Theory

- *Matrices:* Several new advanced algorithms for matrices over $\mathbf{Z}$ or $\mathbf{Q}$ have been implemented (and more will follow in the future). The new algorithms are for computing determinants, characteristic polynomials and minimal polynomials.
- *Modules over Orders:* Modules over orders are now supported.

Algebraic Geometry

- *Schemes:* The algebraic geometry types together with their basic functionality have been moved into the kernel. This includes the integration of many of the specialised curve types such as elliptic and hyperelliptic curves. This significantly increases the power of the geometry types, allowing the user to apply the functionality of both a specialised curve type and the general scheme type to a single object. The change also makes available many generic constructors and standard set and sequence operations, which were previously not available for the geometry types.

- *Schemes:* Maps between schemes have been enhanced with the provision of new constructors and the ability to calculate images and pullbacks available in more circumstances. Maps may now be defined between actual schemes now rather than just between affine or projective spaces.

- *Curves:* Plane curves have been tied very closely to the function field machinery in Magma. The result is that computations with divisors and their Riemann–Roch spaces can be made entirely in the geometric context. The applications include: gap numbers, canonical embeddings of curves, class group computation over finite fields, constructions of geometric codes from curves and many other standard methods in the geometry of curves.

- *Curves:* A new facility for computing with plane conic curves has been written. Its major feature is an implementation of a new algorithm by John Cremona (Nottingham) to find points with reduced coordinates on conics defined over the rational numbers. This implementation is very fast, improving on Cremona's initial test timings.

- *Elliptic Curves:* Elliptic curves are now scheme types and inherit all of the appropriate scheme functionality. In particular, all of the machinery for divisors and differentials now applies to elliptic curves. Subgroup schemes of elliptic curves are now also schemes. John Cremona's database of elliptic curves has been updated to conductor $10,000$.

- *Hyperelliptic Curves:* Machinery for computing the automorphism group of a hyperelliptic curve has been implemented by Michael Stoll. Pierrick Gaudry has implemented a number of methods for counting points on the Jacobian of a hyperelliptic curve. These include the Schoof algorithm for a genus 2 curve. Code for determining Igusa invariants has been provided by Everett Howe while Pierrick Gaudry has implemented an algorithm for constructing a genus 2 curve from the Igusa invariants.

- *Modular Curves:* A package for modular curves has been developed by David Kohel of the Magma group. A modular curve is defined in terms of standard affine modular equations which are stored in precomputed databases. The possible model types are "Atkin", "Canonical", and "Classical".

- *Modular Forms:* A package for modular forms developed by William Stein is now included. Facilities include the computation of a basis of modular forms on $\Gamma_1(N)$, the computation of all newforms of given level, and the decomposition into Eisenstein, cuspidal, and new subspaces.

- *Brandt Modules:* A package for Brandt modules has been developed by David Kohel. Facilities include the construction of a Brandt module on the left ideal class of a definite order in a quaternion algebra over $\mathbf{Q}$, the decomposition of a Brandt module under the action of Atkin–Lehner and Hecke operators and the construction of the Eisenstein and cuspidal subspaces.

- *K3 Surfaces:* A database of K3 surfaces has been added. It contains characteristic data for K3 surfaces embedded in weighted projective spaces in codimension up to 4. The functions used to create the database are also included so that users can extend the database or create similar databases.

- *Handbook:* All geometric chapters of the Handbook have undergone major revision. Many more examples have been included, some of which are extended calculations which illustrate different parts of the new geometry packages working together.

Incidence Structures

- *Graphs:* The `nauty` program due to Brendan McKay for finding automorphisms in graphs has been updated to the latest version (2.0) and its user interface within Magma has been enhanced. This new version of `nauty` is often much faster than the previous version installed within Magma.

- *Graphs:* A database of strongly regular graphs due to Brendan McKay *et al* is now available.

- *Graphs:* A program for the orderly generation of graphs satisfying a range of conditions, written by Brendan McKay, is now accessible from within Magma.

Coding Theory

- *Codes over Fields:* Magma V2.8 incorporates a database containing constructions of the best known linear codes over F_2 of length up to 256. The codes of length up to 36 are optimal. The database is complete in the sense that it contains a construction for every set of parameters. Thus the user has access to $33,152$ best-known binary codes. Similar databases for other small fields will be added in the near future. The implementation of this database has been a joint project with Markus Grassl.

- *Codes over Fields:* Functions are provided to construct Algebraic–Geometric codes.

- *Codes over Rings:* Magma includes facilities for linear codes defined over $\mathbf{Z}_4$ for the first time.

Optimization

- *Linear Programming:* Basic facilities are provided for solving linear programming and integer linear programming problems.

3 Documentation

The Handbook now has bibliographies for each chapter, and much more information concerning the algorithms used has been included in several places.

New chapters in the Handbook for V2.8 are:

- p-Groups
- Generic Abelian Groups
- Automorphism Groups of Groups
- Subgroups of $PSL(2, R)$
- Databases of Groups
- Root Data for Lie Theory
- Groups of Lie type
- Galois Rings
- Ideal Theory and Gröbner Bases
- Modules over Orders
- Rational Curves and Conics
- The K_3 Database
- Modular Curves
- Modular Forms
- Brandt Modules
- Linear Codes over Finite Fields
- Linear Codes over Finite Rings
- Linear Programming

The Categories Overview Document (`catv28.dvi`) has also been greatly expanded.

4 Groups

4.1 General Groups [HB 17]

Changes:

- The function `FPGroup(G:StrongGenerators:=true)` is replaced by `FPGroupStrong(G)`. The former will be withdrawn in a future release, so change your code now!
- The function `ExtraSpecialGroup` has been extended to return both isomorphism types of extra-special group of order p^n.
- The process that produces pseudo-random elements of a group from its generators has been modified to use product replacement with an accumulator. This means two multiplications are required for each step, but results in elements that more closely approximate elements chosen uniformly at random.
- The `IsConjugate` function has been extended to sequences of group elements (thus giving another test for inner automorphisms).

New Features:

- The database of perfect groups compiled by Holt and Plesken has been converted into a new style database.
- A database containing almost simple groups with socle order up to 16 million is available. This database contains automorphism group and maximal subgroup information.

4.2 Permutation Groups [HB 18]

New Features:

- Major improvements to the techniques for computing subgroups using the `Subgroups` family of functions have been introduced. It is now possible to compute subgroups in very much larger groups than before.
- A new very powerful algorithm for finding maximal subgroups in the case of a non-soluble group has been installed. This algorithm, developed by Derek Holt, reduces the problem of finding maximal subgroups to that of knowing the maximals of the non-abelian composition factors (and their automorphism groups).
- The ability to compute maximal subgroups is used as the basis of an algorithm to compute subgroups of small index in a large permutation group (function `LowIndexSubgroups`).
- The function `AutomorphismGroup` determines the full automorphism group of a permutation group.
- Isomorphism of permutation groups may be tested using the function `IsIsomorphic`. Note that this tests for abstract group isomorphism, not permutation isomorphism.
- The function `OrbitRepresentatives` has been introduced to determine just the orbit lengths and representatives; it is more space-efficient than the general-purpose `Orbits`.
- The new function `IsInnerAutomorphism` has been provided.

- The functions `ElementaryAbelianQuotient`, `pQuotient` and `NilpotentQuotient` are now available for permutation groups.

Bug fixes:

- A bug in `DerivedSeries` has been fixed. The bug was responsible for incorrect results produced by `IsSimple` and some other functions.

4.3 General Matrix Groups [HB 19]

Changes:

- The database known as `glnzgps` has been withdrawn as it is superseded by the database of maximal finite subgroups of $GL(n, Q)$.

New features:

- A very efficient algorithm has been implemented for computing the conjugacy classes of elements in a finite matrix group.
- The maximal normal p-subgroup of a finite matrix group may be found using the new function `pCore`.
- The function `ChangeRing` may now be applied to an infinite group. For example, it may be used to change the ring of an infinite group over the integers to a finite group over GF(2), say.
- A restricted version of `LowIndexSubgroups` developed by Leedham-Green and O'Brien is now available for matrix groups.
- The new function `ElementaryAbelianQuotient` returns the p-elementary abelian quotient of a permutation group for a given prime p.
- A database of maximal finite subgroups of $GL(n, Q)$ of degree up to 31 constructed by G. Nebe has been installed (`RationalMatrixGroupDatabase`).
- A database of the finite absolutely irreducible subgroups of $\mathrm{GL}_n(\mathcal{D})$ where $\mathcal{D}$ is a definite quaternion algebra whose centre has degree d over $\mathbf{Q}$ and $nd \leq 10$ has been installed. This collection is due to G. Nebe.

Bug fixes:

- Bug fixes have been made to the functions `CosetTable`, `PCGroup`, `ClassMap` and `IsFinite`.
- Errors in the generators for two exceptional groups of Lie type, namely E_6 and F_4, have been corrected. The errors were discovered by G. Malle.

4.4 Matrix Groups over Finite Fields [HB 19]

New features:

- The function `IsTensorInduced` has been introduced which decides whether or not a matrix group is tensor-induced. It replaces and upgrades `IsSymmetricTensorProduct` which has been withdrawn. The related functions `SymmetricTensorBasis` and `SymmetricTensorPermutations` have been renamed `TensorInducedBasis` and `TensorInducedPermutations` respectively.
- An optional argument has been added to `IsPrimitive` which allows one to search for certain block sizes only; a similar argument added to `IsTensor` allows one to search for factors of particular dimensions.
- The function `OrbitsOfSpaces(G, k)` has been introduced which constructs lengths and orbit representatives for the k-dimensional spaces of the natural vector space under action of G, a matrix group defined over a prime field.
- The function `NumberOfFixedSpaces(g, k)` has been introduced which determines the number of k-dimensional subspaces of the natural vector space fixed under the action of $g \in GL(d, q)$.
- The function `StabiliserOfSpaces` has been introduced which constructs the subgroup of $GL(d, q)$ which stabilises a sequence of subspaces of the natural vector space.
- The function `EstimateOrbit(G, U)`, which estimates the size of the orbit of the subspace U under action of G, has been introduced.
- The function `ApproximateStabiliser(G, A, U)` has been introduced which constructs, using a random approach, a subgroup of the stabiliser of the subspace U under the induced action A of G.
- The permitted arguments to `ExteriorSquare` now include a matrix group.
- The database of irreducible soluble groups has been converted into a new style database.

4.5 Finitely Presented Groups [HB 20, 21]

Changes:

- Coercion of elements of a finitely presented group into a subgroup defined by a set of generating words is now possible to some limited extent. Words in the supergroup which are freely equivalent to one of the defining generators of the subgroup or the inverse of such a generator can be coerced.
- Coercing elements of a finitely presented group into a subgroup defined in terms of Schreier generators is now possible for both unsimplified and simplified presentations.
- Known information about normality of subgroups is stored more consistently, yielding a better performance of functions like `IsNormal`, `Normaliser`, `NormalClosure` and `Core`. The caching of coset tables has been improved in general.
- The functions computing the normal closure of a subgroup were revised and now are able to produce answers in more cases.
- Several functions can now handle finitely presented groups which do not have a presentation assigned, e.g. subgroups defined by a coset table. Among them are `GModule`, `GModulePrimes`, subgroup constructor and normal closure constructor.

- The functions for the representation theory of fp-groups have been revised completely. The new functions `GModulePrimes` and `GModule` are much faster than the old versions and are less restrictive with regard to the maximal size of possible computations. The function `GModule` now returns — like the corresponding functions for other categories of groups — an epimorphism onto the constructed module.
- Some bugs and memory leaks have been fixed. In particular, the system is more tolerant against invalid user input.
- A new version of Éamonn O'Brien's p-quotient algorithm has been installed. In particular, the interactive computation of p-quotients has been improved considerably in this version.
- A new version of George Havas' coset enumerator *ACE* has been installed. Controlled by a wide range of parameters, it enables the user to exert more precise control over the Todd-Coxeter procedure than the old version did. This set of parameters is accepted by various functions which indirectly invoke a coset enumeration, e.g. `Order`, `Index` etc. All functions can handle the old parameters, ensuring backwards compatibility. A global set of parameters, controlling coset enumerations invoked indirectly, e.g. by computing intersections of subgroups, has been introduced. This provides improved performance for many standard operations on finitely presented groups.
- The functions for simplifying a presentation using Tietze transformations have been revised completely. The new version uses an improved simplification strategy and allows more control about the simplification procedure by providing an extended set of parameters. Functions `SetOptions` and `ShowOptions` have been introduced to control the parameter settings of a simplification process. The new function `SimplifyLength` allows partial simplification of a presentation. The elimination of generators is stopped, if further eliminations start to increase the total length of the presentation. This not only may save time, but the resulting presentations in general are more suitable for coset enumeration.
- The functions `Simplify` and `Rewrite` now return an isomorphism from the original group onto the constructed presentation. The group returned is created as a subgroup of the original group, which allows coercing elements from the original group into the new group.
- The function `SchreierGenerators` now by default applies a heuristic method for removing redundant generators. (This feature can be turned off.) Since this reduction is also heavily used in functions internally constructing a generating set of a group, the performance of several functions of the module is thereby improved.
- The function `Order` has been revised completely. New strategies for computing the order of a group or proving its infiniteness are applied. The new function more frequently completes successfully and is in general faster than the old version.

New features:

- Interactive coset enumeration based on the *ACE* program of Havas.
- For homomorphisms with a domain of type `GrpFP`, an attempt is made to compute the kernel, when the kernel is accessed directly or indirectly as a consequence of another function call. This is done by trying to construct a regular permutation representation of the image and, if successful, defining a subgroup of the domain using the obtained coset table. If the kernel can be constructed, computing preimages of subgroups of the codomain is possible.
- The new function `BraidGroup` returns the braid group on a given number of generators as a finitely presented group.
- The new functions `AbelianQuotient` and `ElementaryAbelianQuotient` return the abelian quotient and the p-elementary abelian quotient for a given prime p, respectively, of a finitely presented group.

– The new function `DihedralGroup` can now also construct the infinite dihedral group.

– The new function `PresentationLength` returns the total length of the relators of a presentation.

– The new function `ReduceGenerators` tries to obtain a presentation of a group on fewer generators. It is particularly useful for eliminating redundant generators in subgroups of finitely presented groups obtained as preimages under a homomorphism.

Bug fixes:

– A bug in `Darstellungsgruppe` has been fixed, which was responsible for incorrect answers for groups having an infinite abelian quotient.

4.6 Polycyclic Groups [HB 22]

Changes:

– A new symbolic collector with a much better complexity in the exponents occurring in element vectors has been installed. The speed up can reach several orders of magnitude in typical examples and virtually all computations with polycyclic groups benefit from this improvement.

– The limitation that exponents in a polycyclic presentation and exponents in element vectors be restricted to values less than 2^{30} has been removed.

– A function `PresentationIsSmall` has been added, which enables the user to check whether big integer arithmetic actually is required for a polycyclic presentation. This may be relevant for some category transfers.

– The constructor `PolycyclicGroup` now returns a group in the category `GrpPC` or a group in the category `GrpGPC`, depending on the presentation passed and on the values of a parameter. This simplifies the construction of polycyclic groups and provides a common interface for the closely related group categories `GrpPC` and `GrpGPC`.

– The category transfer functions `FPGroup` and `PCGroup` now check for “small presentations”, since big integers are not supported by the target categories.

– Homomorphisms with a domain of type `GrpGPC` are now checked to be well-defined. (This feature can be turned off.)

– The (trivial) kernels of homomorphisms returned by category transfer functions are now properly embedded in the domain.

– The functions `CosetKernel` and `CosetAction` (if the kernel of the coset action is actually assigned to a variable) now in all cases successfully compute the kernel of the coset action.

– The kernels of homomorphisms with a domain of type `GrpGPC` are no longer computed during the construction of the map, but are computed only when the kernel is actually accessed. This speeds up definitions of homomorphisms and several functions returning or internally using homomorphisms.

New features for general polycyclic groups:

– For homomorphisms with a domain of type `GrpGPC`, an attempt is made to compute the kernel when the kernel is accessed directly or indirectly as a consequence of another function call. Computing kernels currently is possible for codomains of the following data types: `GrpGPC`, `GrpPC`, `GrpPerm` and `GrpAb`, as well as for codomains of the type `GrpMat`, provided that the image is finite.

As a consequence, computing preimages of subgroups of the codomain is now possible in these situations.

- The new function `FreeNilpotentGroup` returns the free nilpotent group of given rank and class as a polycyclic group.
- `FreeAbelianGroup` returns the free abelian group of given rank as a polycyclic group.
- The new function `ElementaryAbelianGroup` returns the elementary abelian group of order p^n for a given prime p and a given integer n as a polycyclic group.
- The functions `AbelianQuotient`, `AbelianQuotientInvariants`, `ElementaryAbelianQuotient` and `FreeAbelianQuotient` provide information about the structure of abelian quotients of a polycyclic group.
- The new functions `FittingSubgroup`, `FittingSeries` and `FittingLength` compute the Fitting subgroup, the Fitting series and the length of the Fitting series of a polycyclic group.
- The new function `UpperCentralSeries` computes the upper central series of a polycyclic group.
- The function `Centre` computes the centre of a polycyclic group.
- `EFASeries` computes a normal series for a polycyclic group, the factors of which are either elementary abelian p-groups or free abelian groups (*elementary or free abelian series; efa-series*).
- The new function `SemisimpleEFASeries` computes a normal series for a polycyclic group G, the factors of which are either elementary abelian p-groups which are semisimple as $\mathbf{F}_p[G]$-modules or free abelian groups which are semisimple as $\mathbf{Q}[G]$-modules (*semisimple efa-series*).
- The new functions `EFAModules` and `SemisimpleEFAModules` return the sequence of $\mathbf{F}_p[G]$- or $\mathbf{Z}[G]$-modules determined by the action of G on the factors of an efa-series of G or on the factors of a semisimple efa-series of G, respectively.
- `EFAModuleMaps` and `SemisimpleEFAModuleMaps` return a sequence of maps from the (non-trivial) subgroups of an efa-series of G or of a semisimple efa-series of G, respectively, onto the $\mathbf{F}_p[G]$- or $\mathbf{Z}[G]$-modules determined by the action of G on the factors of the efa-series or the semisimple efa-series, respectively.
- Four new functions `GModule` compute the $R[G]$-module induced by the action of a polycyclic group G on the maximal abelian quotient ($R = \mathbf{Z}$) or the maximal p-elementary abelian quotient ($R = \mathbf{F}_p$), respectively, of a normal subgroup or of a section of G. The new function `GModulePrimes` computes the set of primes for which the $R[G]$-module induced as described above is non-trivial and the dimensions of the corresponding $R[G]$-modules.

New features for nilpotent polycyclic groups:

- Two new functions `Centraliser` compute the centraliser of an element and of a subgroup of a nilpotent polycyclic group.
- The new function `Normaliser` computes the normaliser of a polycyclic group in another polycyclic group. This function requires the existence of a nilpotent covering group.
- The new function `IsSelfNormalising` tests whether a subgroup of a nilpotent polycyclic group is self-normalising in the supergroup.
- Two new functions `IsConjugate` test whether two elements or two subgroups, respectively, of a nilpotent covering group are conjugate under the action of a subgroup. If so, a conjugating element is computed.

Bug Fixes:

- Some bugs and memory leaks have been fixed. They involve intersection of subgroups, consistency check for presentations, natural epimorphism for quotients, `IsCyclic` and `AbelianGroup`.

4.7 Finite Soluble Groups [HB 23]

New Features:

- A function for computing normal complements (`NormalComplements`) has been implemented.
- The functions `ExtGenerators (G, U)` and `HomGenerators (G, U)` have been introduced to construct explicit generators for Ext $(G/G', U)$, respectively Hom $(H_2(G), U)$, as cocyclic matrices where U is abelian.
- The function `RepresentativeCocyles` has been introduced for computing representative cocycles from G to abelian U as cocyclic matrices.
- The new functions `CentralExtension` and `CentralExtensions` return central extensions of U by G determined by cocyclic matrices.
- The function `CentralExtensionProcess (G, U)` has been introduced for creating a process for forming all central extensions of abelian U by G. Related functions are `NextExtension` and `IsEmpty`.

4.8 Finite p-groups [HB 24]

A new version of the p-quotient algorithm has been installed, and various bugs related to the p-quotient process have been fixed.

The code to construct the standard presentation of a p-group, the automorphism group of a p-group, and to generate descriptions of p-groups has been completely replaced. Much of the existing C code for these computations has been withdrawn and replaced by a package of Magma language functions. The interfaces for these functions have been significantly modified and (hopefully) simplified. Many of the problems in the use of these functions which were related to external file handling should now disappear.

Changes:

- The arguments for `GeneratepGroups` have changed. Many of the optional arguments are no longer supported. The function `Descendants` has been introduced which constructs the descendants [certain central extensions] of a p-group.
- The input argument for the function `StandardPresentation` is now restricted to `GrpPC`; inputs of type `GrpFP` are no longer supported.

New Features:

- The function `StartNewClass` has been introduced which is used to inform the `pQuotientProcess` that the user is about to start a new class computation.
- The function `pCoveringGroup` has been introduced which constructs the p-covering group for a p-group.
- The range of application of `AutomorphismGroup` has been significantly extended.
- The function `ClassTwo` has been introduced which counts precisely the number of p-class 2 p-groups of various orders.
- The function `UnipotentStabiliser`, given as input a unipotent subgroup G of $GL(d, F)$ and U a subspace of the natural vector space, determines the stabiliser in G of U.

- The function `OrderAutomorphismGroupAbelianPGroup` computes the order of the automorphism group of an abelian p-group.
- The function `NumberOfSubgroupsAbelianPGroup` has been introduced which computes the number of subgroups of an abelian p-group.

4.9 Generic Abelian Groups (New) [HB 25]

- A generic (finite) abelian group can be created (using `GenericAbelianGroup`) over any domain provided that an identity and a group operation have been defined.
- Once created one can compute the structure of the group. This is possible if the order of the group is known beforehand. However, it is also possible to compute the structure of the group from a user-supplied set of generators.
- Standard functions for creating subgroups and/or p-Sylow subgroups are provided.
- Standard element operations like computing the order or the discrete logarithm of an element are also provided.
- It is also possible to compute the representation of any group element in terms of a given set of generators of the group.

4.10 Finitely Generated Abelian Groups [HB 26]

New features:

- The new functions `AbelianQuotient` and `ElementaryAbelianQuotient` return the abelian quotient and the p-elementary abelian quotient for a given prime p.

4.11 Subgroups of $PSL(2, R)$ (New) [HB31]

The group $GL_2^+(\mathbf{R})$ of 2 by 2 matrices defined over $\mathbf{R}$ with positive determinant acts on the upper half complex plane $\mathbf{H} = \{x \in C | \mathrm{Im(x)} > 0\}\}$ by fractional linear transformation:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} : z \mapsto \frac{az+b}{cz+d}.$$

Any subgroup Γ of $GL_2^+(\mathbf{R})$ also acts on $\mathbf{H}$. A *fundamental domain* for the action of Γ is a region of $\mathbf{H}^*$ containing a representative of each orbit of the action. Magma contains a package written by Helena Verrill for working with $\mathbf{H}^*$ and with congruence subgroups and their action on $\mathbf{H}^*$. The subgroups of $\mathrm{PSL}_2(Z)$ currently allowed are those of the form $\Gamma_0(N)$, $\Gamma_1(N)$, $\Gamma(N)$, $\Gamma^1(N)$, $\Gamma^0(N)$, and intersections of these groups. The package allows the computation of generators for congruence subgroups, and various other information, such as coset representatives.

Features:

- Computation of generators of congruence subgroups

- Coset representatives for a subgroup of finite index in $\mathrm{PSL}_2(\mathbf{Z})$
- Construction of cusps, cusp widths, and elliptic points of congruence subgroups
- Farey symbols for congruence subgroups
- Action of elements of $\mathrm{PSL}_2(\mathbf{R})$ on the upper half complex plane
- Determination of vertices of a fundamental domain for the action of a congruence subgroup
- Equivalence of points under the action of a congruence subgroup
- Graphics: postscript output of pictures of fundamental domains, points and geodesics, and polygons with geodesic edges (all on the upper half complex plane)

5 Groups of Lie Type

5.1 Root Datum for Groups of Lie Type (New) [HB33]

A data type for root datum has been implemented by Don Taylor and Scott Murray.

5.1.1 Cartan matrices and Cartan types

- Creation of Cartan matrices.
- Identification of Cartan matrices.
- Printing Dynkin diagrams.
- Predicates: IsIrreducible, IsCrystallographic, IsSimplyLaced.
- Computing the vectors in a root system.

5.1.2 Creating Root datum

Any semisimple root datum may be constructed (adjoint, simply connected and everything in between). We plan to add facilities for more general reductive root data in a future release.

5.1.3 Operators on Root Datum

- Basic access functions: (co)root space, simple (co)roots, Cartan matrix, number of positive roots, rank, dimension, Cartan type
- Computing isogeny group
- Dynkin diagram
- Roots are stored in the standard ordering, and are accessed by specifying their position in this order. Also computed in this manner are: coroots, reflection matrices of roots and coroots, and the action on the roots.
- Duals, root subdatum, direct sums.
- Properties: Irreducible, crystallographic, simply laced, adjoint, simply connected.

5.1.4 Related Invariants

- Root norms and root heights.
- Killing forms and dual Killing forms.
- The string through one root in the direction of another.
- The constants defined in Carter for constructing Lie algebras and groups of Lie type: p, q, A (the Cartan integer), N, M, C, η.

5.2 Coxeter Groups [HB34]

Finite Coxeter groups are implemented as a subclass of permutation groups so that they inherit all the operations for permutation groups as well as having many specialized functions. This module was implemented by Don Taylor and Scott Murray.

- Cartan matrix corresponding to a given Dynkin diagram
- Construction of a Coxeter group from a root datum or Cartan matrix
- Dynkin diagram of a Cartan matrix or Coxeter group
- Root datum for a Coxeter group
- Element as a reduced word in the standard generators
- Element of maximal length
- Unique long (short) root of greatest height
- Long word
- Short root of maximal height
- Reflections in Coxeter group
- Reflection subgroup
- Reduced representatives for cosets of the reflection subgroup
- Actions on roots and co-roots
- Coxeter group as a real reflection group
- Coxeter and parabolic subgroups; Transversals
- Braid group, pure braid group and Coxeter group presentation

5.3 Finite Groups of Lie Type (New) [HB35]

We have implemented code for computing in non-groups of Lie type, with elements represented by their Bruhat decomposition. These groups can be defined over any Magma ring. The twisted groups will be implemented in a future release, but for now it is possible to compute with them by treating them as subgroups of the non-twisted groups.

The standard and regular (adjoint) representations for each group may be computed.

- Generators for all classical families of groups of Lie type over a finite field.
- Generators for all exceptional families of groups of Lie type over a finite field.
- Killing form of the Cartan algebra associated with a given Weyl group
- Root elements
- Fundamental roots and their negatives of a simple Lie algebra of given type and rank
- Lie algebra of a Chevalley group as a structure constant algebra
- Adjoint action
- Graph automorphism of a Coxeter group
- Degree of a representation with specified weight
- The BN-pair for a Chevalley group

6 Basic Rings

6.1 Integer Ring [HB 40]

New features:

- An experimental implementation of the number field sieve (NFS) is now available.
- The Cunningham database of factorizations of the form $a^n \pm 1$ due to Brent and te Riele has been updated to the latest version (April 11, 2001).
- The function `KroneckerSymbol(x, y)` now works for $y = 0$, and matches the definition given in Cohen for $y < 0$.
- Many of the basic functions for orders of number fields have been extended to include the ring of integers.

Bug fixes:

- Raising $-\infty$ to the power of $-\infty$ now returns 0 instead of an error.

6.2 Real and Complex Fields [HB 43]

Bug fixes:

- A bug has been fixed that caused a crash when negating real numbers after the default real field had been changed.
- The function `Abs` now works correctly.

6.3 Univariate Polynomial Rings [HB 44]

Polynomials over the integers or rationals are now factored by the exciting new algorithm of Mark van Hoeij (see the Handbook for details and references). The search for the correct combination of modular factors (which has exponential worst-case complexity in the standard Berlekamp–Zassenhaus algorithm) is now performed by van Hoeij's algorithm, which efficiently finds the correct combinations by solving a Knapsack problem via the LLL lattice-basis reduction algorithm.

van Hoeij's new algorithm is *much* more efficient in practice than the original lattice-based factoring algorithm proposed by Lenstra et al.: the lattice constructed in van Hoeij's algorithm has dimension equal to the number of modular factors (not the degree of the input polynomial), and the entries of the lattice are very much smaller. Many polynomials can now be easily factored which were out of reach for any previous algorithm (for example, the Swinnerton–Dyer polynomials).

New features:

- The function `Polynomial` allows the user to create a univariate polynomial without first having to create its parent polynomial ring.

- Automatic coercion has been improved to allow the case where a polynomial over a finite field is added to a rational constant.
- Factorization of polynomials over **Z** or **Q** has been greatly improved by the van Hoeij combination algorithm. Factorization over algebraic number fields has also been improved as a result.
- New function `SwinnertonDyerPolynomial(n)` to create the n-th Swinnerton-Dyer polynomial.
- New function `EqualDegreeFactorization` to compute the equal-degree factorization of a polynomial over a finite field which is known to be a product of distinct degree-d irreducibles alone.

6.4 Rational Function Fields [HB 46]

New features:

- The rational function field type supports the algebraic function fields signatures, see Section 7.4.

6.5 Galois Rings (New) [HB 48]

Magma now provides facilities for computing with Galois rings. The features are currently very basic, but advanced features will be available in the near future, including support for the creation of subrings and appropriate embeddings, allowing lattices of compatible embeddings, just as for finite fields.

Because of the valuation defined on them, Galois rings are Euclidean rings, so they may be used in Magma in any place where general Euclidean rings are valid. This includes many matrix and module functions, and the computation of Gröbner bases. Linear codes over Galois rings will also be supported in the near future.

Features:

- Creation of a default Galois ring (using a default defining polynomial).
- Creation of a Galois ring by a specified defining polynomial.
- Basic structural operations and arithmetic.
- Euclidean operations.

7 Extensions of Rings

7.1 Algebraic Number Fields [HB 53]

Algebraic number fields and their orders have been modified in several ways since Magma V2.7. A new field type has been created to act as the field of fractions of orders of number fields. More functionality has been provided for relative fields and orders and their ideals and quotients by (relative) ideals can now be created.

Removals:

- `ThueSolveInexact`, `BetterPolynomial`, `ClassGroupStructure` and `IsRelative` have all been removed as they were considered to be obsolete. The functionality of each of these can be achieved by alternate means.
- `MinimalInteger` has been removed but its functionality can be achieved by using the function `meet` with second argument the coefficient ring of the order the ideal is of. There are more possibilities for the second argument to this function.
- Polynomials can no longer be coerced into number fields or orders. This removes the ambiguity when coercing polynomials into a ring with a number field or an order as the coefficient ring.
- `NumberField` of cyclotomic and quadratic fields was withdrawn.
- `SetKantVerbose` has been removed but verbose printing can be gained using `SetVerbose`.

General New Features:

- Predicates on orders, fields and ideals: `IsAbsoluteOrder`, `IsAbsoluteField`, `IsAlgebraicField`, `IsNumberField`, `IsSimple`, `IsRamified`, `IsUnramified`, `IsTotallyRamified`, `IsWildlyRamified`, `IsTamelyRamified`, `IsInert`, `IsSplit`, `IsTotallySplit`.

7.1.1 Fields and Orders

Changes:

- `FieldOfFractions` returns a field of the new type `FldOrd`. Some of the functionality of number fields and their elements acting as field of fractions to orders has been transferred to this new type and is no longer present for number fields.
- The parameters `Dedekind`, `Splitting` and `ReducedDiscriminant` have all been removed from `MaximalOrder` and `pMaximalOrder`. `Discriminant`, `Ramification` and `Al` (for algorithm selection) parameters have been implemented for `MaximalOrder`.
- The i-th row of the j-th `MultiplicationTable` now contains the coefficients of the product of the ith and jth basis elements. This used to be the i-th column.
- The spelling of the option `"Padic"` for the `Al` parameter for `GaloisGroup` has been changed to `"pAdic"` to be consistent with spelling throughout the system. `Fast` was renamed into `Conditional`.
- General speed up of class group computations.
- `NumberField(x^2-2)` and `NumberField(x^2-2)` now return *different* fields.
- Improved root finder for embeddings of number fields and computing nth roots of algebraic numbers.

– The matrix returned from `LLL` has been transposed.
– Changed behaviour for creation of fields or orders with non-monic or linear polynomials.
– `Order(RngOrd, AlgMatElt, RngIntElt)`, `ideal<RngOrd| AlgMatElt>` now have the matrix transposed.
– The names of all the `Thue` functions have changed. To create a thue object call `Thue` and to evaluate and solve call `Evaluate` and `Solutions` respectively.
– Relative norm equations will use a faster algorithm now. The new solver will find non–integral solutions for normal extensions.

New Features:

– The type `FldOrd` has been added. Fields of this type are the fields of fractions of an order. They will have the same basis as the order they are the field of fractions of which will not necessarily be a power basis. Elements of these fields have type `FldOrdElt`.
– The type `FldAlg` has been added as an overtype of `FldNum` and `FldOrd` and similarly `FldAlgElt`.
– Non monic polynomials can generate number fields.
– New varargs for number field construction: `Abs`, `DoLinearExtension`, `Global` to allow the creation of non–simple extensions, degree 1 extensions and "global" extensions.
– New signatures for basis functions which also take a ring as well as the order, field or ideal have been implemented returning the basis as elements of the given ring.
– `Embed` can be used to specify the embedding of one algebraic field in another and `EmbeddingMap` will return the existing embedding between two fields.
– Various properties of orders and their elements can be set by calling `SetOrderMaximal`, `SetOrderTorsionUnit` and `SetOrderUnitsAreFundamental`.
– The internal precision used by KANT for calculations in a real field can now also be set directly for fields (not just for orders).
– Solutions of `IndexFormEquations` can be calculated.
– `Completion` of absolute maximal orders at a finite prime
– `pSelmerGroup` of absolute maximal orders at a list of primes.
– `Conductor` of orders, `Different` of orders, elements and ideals.
– `OptimizedRepresentation` will return the homomorphism now, all number field homomorphisms allow for inverses.
– `FactorBasis`, `Relations`, `ClassGroupCyclicFactorGenerators`, `RelationMatrix`
– A parameter for `Regulator` allows access to the current value without first having to compute the fundamental units.
– New (faster) algorithms for factoring monic polynomials over (absolute) maximal orders, embedding of fields.
– `Maximal` printing for orders in *Kash* style. `SetKantPrinting(true);` to print order elements in *Kash* style.
– `Random` for `RngOrd`, `FldAlg`.

- Almost everything not dependent on real arithmetic (essentially class and unit groups) works for relative extensions (extensions of maximal orders).
- `sub<...>` and `Order([ FldAlgElt ])` allow the construction of arbitrary orders.
- Use of the "dot–operator `.`" for orders and their fields of fractions to return the ith basis element as an element of the field.
- Galois theory: `FixedField`, `FixedGroup`. `GaloisGroup` and `Subfields` for (simple) relative extensions.
- Class field theory: a new type `FldAb` for the representation of Abelian extensions of number fields. Supporting functions: `AbelianExtension`, `RayClassField`, `NormGroup`, `Discriminant`, `Degree`, `EquationOrder`, `Conductor`, `AbsoluteDegree`, `BaseField`, `BaseRing`, `CoefficientField`, `CoefficientRing`, `Components`.
- `AbsoluteDiscriminant`, `AbsoluteBasis`.

7.1.2 Elements, Ideals and Quotients

Changes:

- The algorithm for computing roots of elements has been replaced by a faster one.
- The i-th row of `RepresentationMatrix` of an element now gives the coefficients from the multiplication of an element by the i-th basis element instead of the i-th column.
- `Eltseq` for an order element will always return field elements. (This is because in general relative extensions the coefficients have to be non-integral).
- `ideal<O | mat>` now needs a transposed matrix, i.e. the rows of `mat` correspond to the ideal basis, not the columns.
- The codomain of the map returned by `SUnitGroup` is now the field rather than the order.

New features:

- Fractional ideals of orders have been given their own type, `RngOrdFracIdl`. All ideals inherit from this type.
- The functionality for absolute ideals has been provided for relative ideals where possible.
- Quotients of a relative order and an ideal of that order can now be formed. Ideals can be constructed from a module over an order or a matrix and ideals of the coefficient order. This information is used for the basis of the ideal.
- The functions `IsPower`, `Root`, `IsSquare` and `SquareRoot` have been implemented for ideals of orders.
- The `Norm`, `Trace`, `CharacteristicPolynomial`, `MinimalPolynomial` and `RepresentationMatrix` of an element can be found in or over a user specified ring.
- `Divisors` is now available for elements of maximal orders.
- The index of the module $\mathbf{Z}[a]$ in the order containing the element a can be calculated using `Index`.
- `UnitGroup` for absolute maximal orders modulo integral ideals. `RayResidueRings` i.e. `UnitGroups` of orders modulo integral ideals and restriction of the signs of the embeddings. Also provided is `ChineseRemainderTheorem` for infinite places.
- `pRadical`, `MultiplicatorRing` for non primes and relative extensions.

- `ColonIdeal`, `AbsoluteNorm` for ideals, `id meet r`.
- Inverses and division in residue class rings.
- A special algorithm for `PowerProduct` using matrix input.
- `!!` on ideals will create the ideal as an ideal of the ring given.
- Elements can be indexed by integers, for example, `x[i]`, to return the ith coefficient of an element x. The coefficients of an element with respect to a Q-basis can be returned using `Flat`.
- Ramification theory for ideals: `RamificationGroup`, `RamificationField`, `DecompositionGroup`, `DecompositionField`, `InertiaGroup`, `InertiaField`.
- `subset` for ideals. Automorphisms of number fields may be applied to ideals.

Bug fixes:

- All functions returning real conjugates of algebraic numbers now do a precision check, thus the returned values are correct.
- Certain class group computations that failed because of overflow in the real computations work now.
- The move system is more robust wrt. circular references.
- Coercing a polynomial into a ring with an order or number field as the coefficient ring no longer sometimes coerces the polynomial into the coefficient ring but always over the coefficient ring.
- Repeated principal ideal testing without the computation of the generator produced wrong results.

7.2 Quadratic Fields [HB 54]

The quadratic fields and rings have been rewritten to become a part of the algebraic fields and their orders. Quadratic Fields (`FldQuad`) now inherit from the Number Fields (`FldNum`) and Quadratic Rings (`RngQuad`) now inherit from the Orders of algebraic fields (`RngOrd`).

Removals and Changes:

- Some element functions available for a range of discriminants only are now available for one less discriminant (17), (`div`, `mod`, `Modexp`, `Gcd`). These functions along with `Factorization` and `TrialDivision` are only for elements in maximal orders.
- The field of fractions of a quadratic ring is a field of type `FldOrd`. To retrieve the corresponding quadratic field the function `NumberField` must be used.
- Elements are represented with respect to the basis of the order or field which is their parent. For example, `EO.2`, where EO is an equation order of some field where it is different to the maximal order of the field, is $\sqrt{d}$ and EO has basis $\{1, \sqrt{d}\}$. Previously, the elements of such an order were expressed using the basis of the maximal order $\{1, (1 + \sqrt{d})/2\}$.
- `O.1` where O is a quadratic order now returns an element in the field of fractions of O which will be the first basis element of O instead of the second. `O.2` will return the second basis element which was before returned by `O.1`. `Name` will also return the 2nd basis element of O and `AssignNames` will assign the string to this 2nd basis element.

- The algorithm used for `ClassGroup` and `ClassNumber` has changed for small discriminants. The defaults of the parameters have changed for large discriminants resulting in a longer running time and results that are provable under GRH. "ClassGroup" for non maximal orders has been renamed to `PicardGroup` and its size to `PicardNumber`.
- Taking the ith coordinate of an element of a quadratic order `x[i]` returns a rational instead of an integer.
- The defining polynomial of any order is the same as that of its Quadratic field; it does not reflect the presence of the conductor.
- `Regulator` returns a `FldPrElt` instead of a `FldReElt`.
- `NormEquation` returns a sequence containing possibly more than one element as its second return value.
- `BiquadraticResidueSymbol` and `Primary` take arguments of gaussian integer (quadratic ring elements) instead of field elements since an error occurred when field elements were input.

New Features:

- Quadratic fields and their orders are compatible with number fields and their orders.
- Quadratic fields and orders can be created from number fields and their orders using the function `IsQuadratic`.
- All the functionality of the orders and algebraic fields which was absent for the quadratic fields is now present. Some examples are `SplittingField`, `PrimitiveElement`, `AutomorphismGroup` and `GaloisGroup`.
- `NormEquation` is now possible for real quadratic fields.
- A quadratic field may be extended to a relative number field.
- Ideals (both integral and fractional) of quadratic orders may be formed. In addition to the functions for ideals of orders the functions `Conjugate`, `Content`, `Discriminant`, `QuadraticForm` and `Reduction` are available for quadratic ideals. Quadratic ideals can also be created from a quadratic form using the function `QuadraticIdeal`.
- Quotients of quadratic orders by ideals can be taken. The result will have type `RngOrdRes`.
- A `LCM` function has been added for elements of a maximal order.

Bug Fixes:

- A bug in `FundamentalUnit` of a quadratic order has been fixed by the overall changes.
- A bug which resulted in incorrect answers being produced by `BiquadraticResidueSymbol` has been corrected. An example which did not finish running in a reasonable amount of time now does.
- A bug which caused `GCD` computations to crash has been fixed.

7.3 Cyclotomic Fields [HB 55]

The cyclotomic fields have been rewritten to become a part of the algebraic fields and their orders. Cyclotomic Fields (`FldCyc`) now inherit from the Number Fields (`FldNum`). This has resulted in increased speed for most operations and extended the number of available operations greatly.

Removals and Changes:

- `Conjugate(a, n)` now returns a conjugate of a. The signature `Conjugate(a, r)` should be used to retrieve the conjugate of a under the automorphism $\zeta_m \mapsto \zeta_m^n$ where r is ζ_m^n for some n coprime to m, the order of the field.
- The `elt` constructor now takes as many coefficients as the degree of the field instead of an arbitrary number. Coercion will only take sequences of length the degree of the field.

New Features:

- Orders of cyclotomic fields can be found from the field.
- Ideals and quotients of cyclotomic orders can be formed.
- Cyclotomic fields and orders can be extended to relative number fields and orders.

Bug Fixes:

- `RootOfUnity` now returns a root when the order of the field is odd and the exponent given divides twice the order instead of giving an error when the exponent was not equal to twice the order.

7.4 Algebraic Function Fields [HB 57]

Removals and Changes:

- The type of the vector space returned by `RiemannRochSpace`, `Module` and `Relations` has changed.
- The order of the arguments to `Module` and `Relations` has changed.

New Features:

- `FunctionField(g)` where g is a bivariate polynomial.
- For all algebraic function fields `RationalExtensionRepresentation`, `ExactConstantField` and `SeparatingElement` have been added.
- The following new functions apply to function field and/or order elements: `IsSeparating`, `Root`, `IsConstant`, `IntegralSplit`, `Expand`, `RationalFunction`, `ProductRepresentation` and `Minimum`. `Random` elements of global function fields and their orders are now obtainable.
- The following new functions apply to divisors and/or places: `Gcd`, `Lcm`, `IsCanonical`, `ShortBasis`, `ComplementaryDivisor`, `NumberOfSmoothDivisors`, `Roots`, `RamificationDivisor`, `GapNumbers`, `WronskianOrders`, `WeierstrassPlaces` and `IsWeierstrassPlace`. The function `Divisor` has a new signature that takes two ideals belonging to maximal orders of a function field.
- An algorithm for computing the divisor class group of a global function field has been implemented. As a consequence the following functions relating to the class group may be applied to a global function field: `ClassGroup`, `ClassGroupAbelianInvariants`, `ClassNumber`, `PrincipalDivisorMap`, `ClassGroupExactSequence`, `GlobalUnitGroup`, `IsGlobalUnit`, `SClassGroupAbelianInvariants`, `IsGlobalUnitWithPreimage`, `SClassNumber`, `IsSUnitWithPreimage`, `IsSPrincipal`, `SUnitGroup`, `SClassGroupExactSequence`, `IsSUnit`, `SPrincipalDivisorMap`, `SRegulator`, `ClassGroupPRank`, and `HasseWittInvariant`.
- The following functions relating to the class group may be applied to maximal finite orders and their ideals in a global function field: `ClassGroup`, `ClassGroupAbelianInvariants`, `ClassNumber`, `ClassGroupExactSequence`, `PrincipalIdealMap`, `IsPrincipal` `UnitGroup`, `IsUnitWithPreimage`.
- The function `WeilRestriction` is available for elliptic function fields.
- Machinery for working with differentials (`DiffFunElt`) is now available. In addition to basic arithmetic the most relevant functions are `DifferentialSpace`, `DifferentialBasis`, `Differentiation`, `Differential`, `Divisor`, `SpaceOfDifferentialsFirstKind`, `BasisOfDifferentialsFirstKind`, `SpaceOfHolomorphicDifferentials`, `BasisOfHolomorphicDifferentials`, `Residue`, `Valuation`, `IsCanonical`, `FunctionField`, `IsExact`, `Cartier`, and `CartierRepresentation`.
- The modules returned by `RiemannRochSpace`, `SpaceOfHolomorphicDifferentials`, `Relations`, `Module` and `DifferentialSpace`, come with embeddings into a function field or space of differentials of a functions field. The intersection, sum etc. of such modules is possible as well as coercion between the field or space and the module.
- `Factorization` of polynomials (univariate and multivariate) over function fields and their orders can now be achieved.

7.5 Newton Polygons [HB 58]

Removals and Changes:

- The interpretation of a precision with respect to some exponent denominator has been clarified. The exponent denominator it is with respect to is the lowest common multiple of the exponent denominator of the root or expansion and the exponent denominators of the coefficients of the polynomial. Since denominators have if anything increased, the precision of answers has decreased.
- The curve taken as an argument to `NewtonPolygon` must now be of the type `Crv`—one of the new scheme types.

Bug Fixes:

- Some expansions giving more precision than was asked for have been amended, reducing the precision to the required amount. (This excess was above that due to the reinterpretation of precision with respect to exponent denominators).

7.6 Local (including p-adic) Rings and Fields [HB 42, 59]

Bug Fixes:

- A problem with `SuggestedPrecision` has been fixed.
- `UnramifiedExtension` where $p^f > 2^{30}$ is now possible.
- A bug involving resultants of multivariate polynomials has been fixed.
- The error checking on `HenselLift` has improved.
- An improvement in the linear algebra over local rings has been attempted.

8 Commutative Algebra

8.1 Ideal Theory and Gröbner Bases (New) [HB 50]

Magma now provides facilities for computing with Gröbner bases of ideals of polynomial rings over Euclidean rings (including the important case of the integer ring **Z**). Such Gröbner bases are computed in Magma by an extension, due to Allan Steel, of Jean-Charles Faugère's F_4 algorithm which uses sparse linear algebra.

The valid Euclidean rings in Magma supported are: the integer ring **Z**, the integer residue class rings $\mathbf{Z}_m$, the univariate polynomial rings $K[x]$ over any field K, Galois rings, and valuation rings.

The extension of Faugère's algorithm depends on an algorithm for computing a unique echelon form of a sparse matrix over a general Euclidean ring. Based on this new sparse matrix algorithm and some other techniques, Magma ensures that a Gröbner basis over a Euclidean ring is reduced, and **unique** (see the Handbook for details). Uniqueness is even ensured for rings with zero divisors!

Many of the standard functions based on Grobner bases over fields also carry over to ideals defined over Euclidean rings. One can even effectively compute with more general rings which are not Euclidean (see the Handbook examples).

New features:

– Ideals may be defined over general Euclidean rings, and Gröbner bases of such ideals may be computed.

8.2 Affine Algebras [HB 51]

New features:

– Affine algebras may now be created whose base ring is a general Euclidean ring.

8.3 Modules over Affine Algebras [HB 52]

New features:

– Modules over affine algebras may now be created whose base ring is a general Euclidean ring. Most current module operations (for modules over affine algebras over fields) are also valid for such modules, including the computation of syzygy modules.

9 Linear Algebra and Module Theory

9.1 Matrices [HB 62]

New Features:

- The algorithm to compute determinants of matrices over **Z** or **Q** has been greatly improved by first using the sparse Smith–algorithm techniques, and then using the modular algorithm of Abbott, Bronstein and Mulders.
- A new algorithm of Allan Steel for computing minimal and characteristic polynomials of matrices over **Z** or **Q** has been implemented. This is based on the p-adic nullspace algorithm and so is very much faster than the previous algorithm.
- New function `MinimalAndCharacteristicPolynomials` to compute the minimal and characteristic polynomials simultaneously (faster over some rings than calling the separate functions), and new functions `FactoredMinimalAndCharacteristicPolynomials`, `FactoredMinimalPolynomial`, and `FactoredCharacteristicPolynomial` which are faster over some rings than calling `Factorization` on the minimal or characteristic polynomials.
- New function `AntisymmetricMatrix` to create an antisymmetric matrix.

9.2 Modules over Orders (New) [HB 65]

Modules over orders of type `ModOrd` exist only for maximal orders so that the modules are over a Dedekind domain for which the structure theory is greater.

New Features:

- The `Module` function can create modules over orders from orders, vectors, ideals and tuples of vectors and ideals.
- Submodules and quotients of modules by submodules can be created using the `sub` and `quo` constructors. For quotient modules a generalised `SmithForm` can be computed.
- The `BaseRing`, `Degree`, `NumberOfGenerators`, `Determinanant`, `Dimension` and the vectors generating a module can all be retrieved from a module.
- Equality of modules, whether an element lies in a module and whether one module is a subset of another can all be determined. Intersections of modules can also be taken.
- Modules can be multiplied with ideals and added. Modules can also be formed as the product of a module element and an ideal.
- The `Basis` of a module can be determined as well as its `ElementaryDivisors`.
- The `Dual` of a module wrt. scalar products can be computed.
- The `SteinitzClass` and `SteinitzForm` of a module can be computed.
- Homomorphisms between modules can be created as well as the hom–module of homomorphisms between two modules. The image and kernel of homomorphisms can be calculated. A module can be tested for being a submodule of another. The morphism of a module into a submodule or quotient module is returned by `Morphism`.
- Elements of Modules can be created by coercion of a sequence, vector or module element into a module. These elements can be added, subtracted, multiplied and divided by scalars and multiplied by ideals. Equality of module elements can be determined and elements can be represented as sequences.

10 Lattices and Quadratic Forms

10.1 Lattices [HB 66]

New features:

– Many lattices in the database of lattices maintained by Neil Sloane are now available in a Magma database (`LatticeDatabase`). Lattices already available in Magma through standard intrinsic functions are not included in this database.

10.2 Binary Quadratic Forms [HB 67]

Removals and Changes:

– The Shanks algorithm is now used in place of the Gauss algorithm for the cases in which composition of forms is specified using the `*` and `^` operators. The Gauss algorithm is the default when when composition of forms is specified using the functions `Power` and `Composition`.

– The function `IsOne` has been renamed to `IsIdentity`.

Bug Fixes:

– The parameter `Reduction` is now provided for `Composition` and `Power`. Its default value is `false`.

11 Representation Theory

11.1 Modules over Matrix Algebras [HB 76]

New Features:

– Given a $K[G]$-module M, the new function `GModuleAction` returns the action of G on M as homomorphism from G into the matrix group $\mathrm{GL}_n(K)$.

11.2 Characters of Finite Groups [HB 77]

Bug fixes:

– The `Symmetrization` function has been altered to reverse the outcomes of $[1, 1]$ and $[2]$. This gives the more usual meanings to these symmetrizations.

– Conjugation of a character of a normal subgroup is repaired.

– A bug in `OrthogonalComponents` and `SymplecticComponents` which led to incorrect deductions about character irreducibility has been rectified.

12 Algebraic Geometry

For version 2.8 the algebraic geometry suite has been completely re-engineered. Most functionality has been moved into the kernel and idiosyncrasies of the first geometry implementation have been removed. The major categories that have been added or enhanced are

- Schemes
- Rational scrolls
- Zero-dimensional schemes
- Algebraic curves
- Divisor groups of plane curves
- Rational curves
- Elliptic curves
- Hyperelliptic curves

12.1 Schemes [HB 81]

There are three major enhancements visible to the user. They all arise as part of the re-implementation of the geometry types. The first is that with the types now part of the Magma kernel, objects interact more effectively with other parts of Magma. For example, it is now possible to have sets of points.

The second is the new point types allow coordinates of points to lie in rings other than the base ring of the scheme. This is especially useful for schemes defined over fields, since the field extension machinery is now tied more closely to the Gröbner basis machinery for operations like finding the singularities of a curve.

The third is far more powerful mapping types, including new constructors and more familiar usage. Maps can now be defined between arbitrary schemes (with some conditions on the base ring when Gröbner basis techniques are required).

Beyond these, there are a number of new object types.

New Features:

– More flexible creation of schemes defined over a ring k

– Points of schemes with coefficients in k-algebras

– New map constructors

– Enhanced image and preimage calculation under maps

– A new type for ruled surfaces and other rational scrolls

– A new type for products of affine and projective spaces

12.1.1 Linear Systems

The linear systems module has also been moved into the kernel. Linear systems can now be created on both affine and projective spaces rather than just projective. The following changes were made.

- The comparison maps between `CoefficientMap(L)` and `PolynomialMap(L)` for a linear system L are now returned as maps which take an argument x rather than these intrinsics taking x as a second argument.
- The various intrinsics `Subsystem(L,X)` where L is a linear system and X is some other data have been renamed `LinearSystem(L,X)`.

12.2 Algebraic Curves

All of the distinct types of curve in version 2.7 have been translated into the new scheme types. In particular, the functions and usage for different kinds of curves is converging to a standard. Most of the new curve functionality is based on our new function field machinery, but in the context of curves the results are available directly without function field knowledge.

12.2.1 Plane Curves [HB 82]

The new types are especially powerful for our existing curve types. Although it is hard to list particular features, this is exemplified by the computation of points on elliptic curves. Since points are now allowed over any extension of the base ring, one can now work with points over a variety of rings on the same elliptic curve without having to create a new curve over each extension.

- Transfer of all curve types to the new regime resulting in more uniform functionality.

12.2.2 Rational Curves and Conics [HB 84]

- New data type for rational curves.
- A parametrisation algorithm for rational curves given a rational point.
- A canonical parametrisation algorithm reducing rational curves to conics irrespective of the existence of a point.
- New data type for plane conics.
- A new algorithm for finding a point with small coefficients over the rationals if one exists.
- Type translation from curves of genus 0 to rational curves.

12.2.3 Function Fields and Divisors on Curves [HB 82]

- The computation and arithmetic of the function field of a curve has been more tightly associated to the curve itself.
- New data type for places and divisors on curves; functionality is retrieved from the associated function field.
- Computation of the divisor class group for curves defined over finite fields.

12.3 Elliptic Curves [HB 85]

Elliptic curves are now part of the general scheme machinery. It is highly recommended to read the appropriate chapters in the Handbook for a better description of how these changes affect the use of elliptic curves.

12.3.1 General Elliptic Curves

Removals and Changes:

- Elliptic curves are now scheme types (of type `CrvEll` still), and inherit all of the appropriate scheme functionality. As part of this change, the curve is no longer the parent of points but various *point sets* are instead. The type of a point is now `PtEll` and the type of a point set is `SetPtEll`.
- Subgroup schemes of elliptic curves are now also schemes, of type `SchGrpEll`. All functions which apply to subgroup schemes should also apply to the elliptic curves.
- Isomorphisms are not compatible with isogenies at the moment.
- In keeping with the scheme philosophy, `BaseExtend(E, K)` does not work unless K is an extension of the base field of E. In particular, this will fail if K is a finite field and E is defined over $\mathbf{Q}$. The intrinsic `ChangeRing` can be used instead in this case.
- Subschemes of elliptic curves are now constructed using scheme functions, and are no longer a special type.
- The deprecated function `EllipticCurve(K, j)` has been removed; use `EllipticCurve(K!j)` instead.
- The deprecated function `IsPoint(S, E)` has been removed; use `IsPoint(E, S)` instead.
- The deprecated function `IsOrderOfPoint` has been removed; use `IsOrder` instead.
- The functions `mTorsionSubgroup`, `nTorsionSubgroup`, and `pTorsionSubgroup` have been removed; use `TorsionSubgroupScheme` instead.
- The functions `Lift(E, K)` and `Lift(E, K, m)` have been removed; use `BaseExtend` or `ChangeRing` instead.
- The function `Subgroup` has been renamed to `SubgroupScheme`.
- Since elliptic curves are also (trivial) subgroup schemes, the function `DefiningPolynomial` for a subgroup scheme has been renamed to `DefiningSubschemePolynomial` to make it clear which function is meant.
- The functions `WeierstrassCoefficients(E)` and `RationalPoints(E, K)` should be considered deprecated and will be removed in a future release.

New features:

- Isomorphisms between elliptic curves are now scheme maps, and inherit the appropriate functionality of these maps.
- The function `PointsAtInfinity` has been added for symmetry with the hyperelliptic curves.
- Elliptic curves can be created from general curves which are already in the correct form, without the need to specify a point on the curve.
- The functions `IsIsomorphic` and `SimplifiedModel` now work for curves defined over any field which supports root finding, not just finite fields.
- The function `Isomorphism` has been added to return the isomorphism between two curves that are known to be isomorphic.
- The functions `IsogenyFromKernel` and `IsogenyFromKernelFactored` are no longer restricted to elliptic curves defined over $\mathbf{Q}$ or a finite field.

12.3.2 Elliptic Curves over the Rational Field

New features:

- Improvements have been made to the rank computation in the no two-torsion case.
- John Cremona's database of elliptic curves has been updated to conductor 10 000.

Bug fixes:

- A bug has been fixed in `CremonaReference` which would occasionally cause an incorrect reference to be returned.
- The computation of the torsion subgroup no longer crashes or returns incorrect results for curves with large coefficients.
- Various crashes in the rank computation have been fixed.

12.3.3 Elliptic Curves over Finite Fields

New features:

- An early abort feature has been added to the point counting intrinsic `SEA`, which enables the computation to be terminated if the order is divisible by too many small primes.

Bug fixes:

- A bug has been fixed in `IsIsomorphic` that caused an incorrect isomorphism to be returned when used on supersingular elliptic curves in characteristics two or three.

12.4 Hyperelliptic Curves [HB 86]

12.4.1 Hyperelliptic Curves

New Features:

- New intrinsics for finding automorphisms and for isomorphism testing have been added (`AutomorphismGroup` and `IsIsomorphic` resp.).
- New intrinsics that change the model of a hyperelliptic curve are now available. Those are `SimplifiedModel`, `pMinimalModel`, `pMinimalWeierstrassModel`, `ReducedModel`, `pNormalModel` and `MinimalWeierstrassModel`,
- Using an algorithm due to Mestre, one can construct a curve of genus 2 from a given a set of Igusa-Clebsch invariants (the function `CurveFromICInv`). The implementation of the algorithm must be credited to P. Gaudry. The reduction of the curve so obtained is achieved using ideas due to P. Van Wamelen.
- The function `Frobenius` applies the Frobenius to a point on a hyperelliptic curve.

12.4.2 Jacobians of Hyperelliptic Curves

New Features:

- Two new `Order` functions are provided to compute the order of a point P on a Jacobian J where the order of P or of J is bounded and where (optionally) some modular information is known about that order. Being able to use this extra information dramatically improves the running time of the `Order` routines. These are due to P. Gaudry.
- The function `Frobenius` applies the Frobenius to a point on a Jacobian.
- The function `WeilPairing` computes the Weil pairing of two points on a Jacobian defined over a finite field.
- The `Order` function for Jacobians over finite fields has been rewritten by P. Gaudry. Jacobians defined over a curve of genus 2 are treated separately. More generally some specific algorithms are provided to deal with some special cases.

12.5 Modular Curves (New) [HB 87]

A package for computing with modular curves has been developed by David Kohel. Modular curves in Magma are a special type of plane curve. A modular curve X is defined in terms of standard affine modular equations which are stored in precomputed databases. Those modular curves presently available are defined by modular equations—a bivariate polynomial relation between the j-invariant and one of several standard functions on $X_0(N)$. These give singular, affine models for $X_0(N)$ designed for computing isogenies of elliptic curves.

Features:

– Creation of a modular curve of specified level rom a database. Possible model types are *Atkin*, *Canonical* and *Classical*.

– Database of modular equations: *Atkin*, *Canonical* and *Classical*.

– Parametrization of the isogenies of an elliptic curve by points on some $X_0(N)$.

– j-invariant of a modular curve over a field.

– Base curve of a modular curve.

– Automorphisms: Atkin–Lehner involution.

– Hilbert and Weber class polynomials.

12.6 Modular Symbols [HB 88]

New features:

– The functions `CuspidalSubspace` and `NewformDecomposition` have been *very* greatly sped up, using a variation on the new minimal and characteristic polynomial algorithms.

12.7 Brandt Modules (New) [HB 89]

Brandt modules provide a representation in terms of quaternion ideals of certain cohomology subgroups associated to Shimura curves $X_0^D(N)$ which generalize the classical modular curves $X_0(N)$. The Brandt module datatype is that of a Hecke module – a free module of finite rank with the action of a ring of Hecke operators – which is equipped with a canonical basis (identified with left quaternion ideal classes) and an inner product which is adjoint with respect to the Hecke operators.

Features:

- Construction of a Brandt module on the left ideal class of a definite order in a quaternion algebra over $\mathbf{Q}$.
- Arithmetic operations with module elements.
- Inner product of elements with respect to the canonical pairing on their parent.
- Elementary invariants of a Brandt module: Level, discriminant, conductor etc.
- Decomposition of a Brandt module under the action of Atkin–Lehner and Hecke operators.
- Eisenstein subspace, cuspidal subspace.
- Operations on subspaces: Orthogonal complement, intersection.
- Properties of subspaces: Eisenstein, cuspidal, decomposable.
- Construction of Hecke and Atkin Lehner operators.
- q-expansions associated with a pair of elements of a Brandt module.
- Determination of the dimension of a Brandt module of given level obtained using standard formulae.

12.8 Modular Forms (New) [HB 90]

A package for computing with modular forms has been developed by William Stein.

Features:

- Computation of a basis of modular forms on $\Gamma_1(N)$ of any integer weight greater than one.
- Computation of all newforms of given level.
- Computation of all reductions of a newform of given level modulo a prime.
- Embeddings of a newform of given level in the complex and p-adic fields.
- Arithmetic with modular forms.
- Characteristic polynomials of Hecke operators.
- Decomposition into Eisenstein, cuspidal, and new subspaces.
- Dimension formulas.

12.9 K3 Surface Database [HB 91]

- Raw data for the K3 database in codimension at most 4.
- Functions for interrogating the database.
- Functions for modifying the database in light of new geometrical constructions.

13 Incidence Structures

13.1 Graphs [HB 93]

Changes:

- The `nauty` program (B. McKay) has been updated to version 2.0. Version 2.0 has a larger range of invariants to work with when trying to refine a graph's partition, thus considerably speeding up the search for a canonical labelling of difficult graphs. The available invariants and their respective use are fully documented in the Magma handbook.

  In addition, Version 2.0 now uses dynamic memory allocation which allows `nauty` to work on graphs on any size.
- In addition to `AutomorphismGroup`, the functions `IsDistanceTransitive`, `IsEdgeTransitive`, `IsTransitive`, `IsPrimitive`, `IsSymmetric`, `EdgeGroup`, `OrbitsPartition`, `IsIsomorphic`, and `CanonicalGraph` now allow users to set a parameter suite given them more control over which automorphism group to compute.

New Features:

- A function `TestNautyInvariant` is provided to help the user decide which invariant is to be used to run `nauty`.
- The function `StronglyRegularGraphsDatabase` gives access to a catalogue of strongly regular graphs due to B. McKay *et al.* Standard access functions to help retrieve the graphs in the catalogue are also provided.

  The function `GenerateGraphs` gives access to the graph generation program written by B. McKay. This feature is only available on Unix platforms as graphs are generated (and read from) within a Unix pipe.

  The graph generation program allows users to generate any isomorph-free collection of graphs responding to several criteria like, among others, the connectedness of the graphs, or whether they are 3 or 4-cycle free.
- The function `OpenGraphFile` enables the user to open any file of graphs or any graph generating pipe. The graphs must be in a specified format, and the pipe mechanism is only available on Unix platforms. The function `NextGraph` is used to read any graph in the stream output from `GenerateGraphs` or `OpenGraphFile`.

14 Coding Theory

14.1 Linear Codes over Finite Fields (New) [HB 97]

Changes:

– The function `VerfiyMinimumWeight` was not performing well. It has been replaced by the function `VerifyMinimumWeightLowerBound` which determines whether or not a given value is a lower bound of a given code. The function `VerifyMinimumWeightUpperBound` performs similarly for an upper bound.

– The function `SubfieldRepresentation` has been replaced by `SubfieldRepresentationCode` along with the subfield code constructions `SubfieldRepresentationParityCode` and `SubfieldCode`.

– The old function `ExpurgateCode` now allows multiple words to be expurgated, while `ExtendCode` now allows an extension of arbitrary length, and `DirectSum` now takes a sequence of arbitrary length. The old function `VectorSpace` is now also named `RSpace`.

New Features:

– The advanced Zimmermann algorithm for `MinimumWeight` has been optimized: it now discards information sets of low rank. A new parameter `MaximumTime` has been introduced to abort lengthy calculations, and the parameter `cutoff` has been renamed `RankLowerBound`.

– A new database gives the user access to the best known binary linear codes of length up to 256 (joint project with Markus Grassl). Codes can be accessed through the functions `BestKnownLinearCode`, `BestDimensionLinearCode`, `BestLengthLinearCode`, and their abbreviations `BKLC`, `BDLC` and `BLLC` respectively. Bounds can be accessed through the functions `BKLCLowerBound`, `BKLCUpperBound`, etc. The functions are designed so that any two of the length, dimension, or minimum weight can be specified, and the optimal result for the missing parameter is returned.

– New constructions for codes are `CordaroWagnerCode`, which returns the dimension 2 Cordaro–Wagner code, `MDSCode`, which returns a maximum-distance-separable code, `QuasiCyclicCode` which returns the quasicyclic code formed by concatenating cyclic codes, and `NonPrimitveAlternantCode` which returns a non-primitive alternant code over GF(2).

– Some new constructions based on circulant matrices and related to quadratic residue codes are `DoublyCirculantQRCode` and `BorderedDoublyCirculantQRCode`. The function `TwistedQRCode` returns a twisted quadratic residue code. A generalisation `PowerResidueCode` returns the code of an arbitrary power residue.

– New functions which take a code and produce a new code are: `ResidueCode` which performs a single Griesmer step, `PadCode` which extends each codeword with zeros, `ConstructionY1` which performs construction Y1, and `ExpurgateWeightCode` which expurgates all codewords of given weight.

– New functions which take multiple codes to produce new codes are: `CodeComplement` which returns the complement subspace from a given subcode, `CodeJuxtaposition` which joins the generator matrices of two codes, `'cat'` which concatenates all combinations of codewords from two codes, and `ZinovievCode` which creates generalised concatenated codes. Further such functions include `ConstructionX`, `ConstructionX3` and `ConstructionXX`.

– New functions relating to subcodes are: `Subcode` which returns a subcode of given dimension, and `SubcodeBetweenCode` which returns a code nested between an existing code-subcode pair.

- The new function `SubcodeWordsOfWeight` returns the subcode generated by codewords of specified weights.
- The new function `WordsOfBoundedWeight` returns all code words whose length lies between the given bounds.
- Machinery has been provided which allows the construction of algebraic–geometric codes. See functions `AlgebraicGeometricCode` and `AGCode`.

14.2 Linear Codes over Finite Rings (New) [HB 98]

Magma now has facilities for linear codes defined over $\mathbf{Z}_4$. These facilities are currently basic but will be extended in the near future. Codes over general finite rings (including Galois rings) will also be supported in the future.

Features:

- Various generic constructions and functions for $\mathbf{Z}_4$-codes.
- Construction of cyclic codes from factorizations of $x^n - 1$ over $\mathbf{Z}_4$.
- Standard form of a $\mathbf{Z}_4$-code, giving the abelian group structure.
- The Gray map and a function to test whether the image of a $\mathbf{Z}_4$-code under the Gray map is linear.
- Minimum weight and weight distribution.
- Weight enumerators: complete, symmetric, Lee and Hamming.
- Construction of Kerdock and Preparata codes.

15 Optimization

15.1 Linear Programming (New) [HB 100]

Basic facilities are now provided for linear programming.