Summary of New Features in Magma V2.4

December 14, 1998

1 Introduction

This document provides a terse summary of the new features installed in Magma for release version V2.4 (December 14, 1998). Previous releases of Magma are: 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); release notes for these versions are found in the document `relprev.dvi`.

2 Summary

- Two important group theory databases are now included: the Besche-Eick database of all groups up to order 1000 (excepting orders 512 and 768) and the Hulpke database of transitive groups of degree up to 22.

- A facility for analyzing matrix groups of large degree defined over finite fields has been installed. This module uses the Aschbacher classification of maximal subgroups of $GL(n, q)$. The module was written by Derek Holt, Alice Niemeyer, Eamonn O'Brien and Anthony Pye.

- The Schönhage-Strassen FFT-based algorithm for the multiplication of very large integers and an asymptotically-fast algorithm for the division of large integers has been implemented.

- Two separate FFT-based algorithms are now employed for the multiplication of univariate polynomials over the integers and similar coefficient rings. Also, an asymptotically-fast algorithm for the division of polynomials has been implemented and modular exponentiation is now performed using pre-inversion of the modulus.

- A new efficient packed representation for polynomials over GF(2) has been implemented. This has led to dramatic speed-ups for all computations within finite fields of characteristic 2. A database of sparse irreducible polynomials over GF(2) has also been constructed for all degrees up to 11000.

- The Shoup algorithm for factorization of polynomials over finite fields has been implemented, leading to very significant speed-ups.

- Stage 1 of a major revision and extension of the power series module has been completed. In this stage, the existing machinery has been redone in greater generality and asymptotically-fast algorithms are employed for multiplication and division. Series with fractional exponents are now permitted.

- The KANT number field machinery corresponding to Kash 1.9 has been installed. This is the first major upgrade of the Magma general number field facility since Kash V1.5 in 1995. This version of KANT provides the Magma user with greatly enhanced performance for many fundamental algorithms. Particularly noteworthy is a Round 4 integral basis algorithm and a much improved class group algorithm.

- Major new number field facilities available in Magma V2.4 (courtesy of KANT) are ray class groups, S-unit groups, prime ideal decompositions of fractional ideals, subfield lattices, automorphism groups of normal and abelian extensions, solution of Thue equations, unit equations and index form equations.

- A set of functions have been provided for computing q-expansions of the standard elliptic and modular functions. These include the Weierstrass $\wp$-function, Eisenstein series, Dedekind η-function, Jacobi θ-function, elliptic j-invariant and discriminant function.

- The facilities for elliptic curves have undergone major expansion. Noteworthy is the introduction of general machinery for working with isomorphisms, isogenies and rational maps between curves. The new machinery allows comparatively easy computation of things such as the endomorphism ring of a curve over a finite field and the eigenvalues of the Frobenius automorphism (the latter being useful in point-counting).

- The Cremona database of elliptic curves having conductor up to 5300 has been installed.

- The design of the facility for finite planes has been revised, firstly, to make it easier to work with planes and their subplanes and secondly, to make it follow similar conventions to other structures.

3 Removals and Changes

- The libraries directory of Magma has been renamed from `LIBS` to `libs`—this is now consistent with all the other Magma filenames, which are in lower case.
- The following libraries of groups have been replaced by databases:
  - The soluble groups of order up to 100 (`gps100`);
  - The library of 2-groups of order dividing 256 (`twogps`);
  - The library of 3-groups of order dividing 729 (`thrgps`);
  - The library of primitive groups (`prmgps`);
  - The library of transitive groups (`trngps`).

  The use of databases means that the information may be accessed directly without first having to load a library. In general, the new databases offer equivalent or extended functionality compared to the old libraries. However, users wishing to continue using any of these libraries may obtain them by mailing `magma@maths.usyd.edu.au`.
- The function `IsDivisible` has been renamed to `IsDivisibleBy`.
- For consistency, the function `SquareFree` has been renamed to `Squarefree` and the function `SquareFreeFactorization` has been renamed to `SquarefreeFactorization`; the old function names have been removed from the documentation but kept in the executable to allow old code to continue to work for this release. ("Squarefree" seems to be more standard than "square-free"—Knuth, for example, uses the former.)
- The function `ChineseRemainderTheorem` or `CRT` taking 3 integers has been removed to avoid confusion. (`Solution` still retains the original functionality.)
- The function `ChineseRemainderTheorem` or `CRT` taking 3 integer sequences has been changed so that it now only takes 2 integer sequences (the original "multipliers" sequence is now considered to be all ones). (`Solution` still retains the original functionality.)
- The function `PartialFractionDecomposition` now produces the full partial fraction decomposition of a rational function. The original functionality is now given by the function `SquarefreePartialFractionDecomposition`.
- The function `Variety` now returns a sequence of tuples.
- The functions `PowerSeriesAlgebra` and `LaurentSeriesAlgebra` have been removed from the documentation in anticipation of their withdrawal in a future release.
- The number field function `BetterPolynomial` has been replaced by `OptimizedRepresentation` which has slightly different semantics. The old function will be retained for compatibility in the medium term.
- All number field functions that return (inexact) real or complex numbers have been changed so that instead of returning fixed precision numbers, they return free numbers (type FldPrElt).
- The category names for formal series have been changed (and new ones have been added as well).
- The category name for elliptic curves has been changed to `CurveEll` and the category name for points on an elliptic curve has been changed to `CurveEllPt`.

– The following incidence structure intrinsics have changed to accept the incidence structure in which the operation should be performed as the first argument: `Block`, `BlockDegree`, `BlockSize`, `ConnectionNumber`, `Covalence`, `IsParallelClass`, `Line`, `PointDegree`.

– The function `IsBlock(S, D)`, where D is an incidence structure and S is a set of points of D, is now `IsBlock(D, S)`.

– The following plane intrinsics have changed to accept the plane in which the operation should be performed as the first argument: `AllPassants`, `AllSecants`, `AllTangents`, `CentralCollineationGroup`, `Conic`, `ContainsQuadrangle`, `Coordinates`, `Exterior`, `ExternalLines`, `Index`, `Interior`, `IsArc`, `IsCollinear`, `IsComplete`, `IsConcurrent`, `IsParallel`, `IsUnital`, `Knot`, `ParallelClass`, `Pencil`, `QuadraticForm`, `Support`, `Tangent`, `UnitalFeet`.

4 Language and System Features [HB 1–5]

New features:

– New operator `cmpne` which is the logical NOT of the operator `cmpeq` (to allow equality testing of arbitrary objects).

– The procedure `SetColumns` now checks whether its argument is in range (for the first time!).

– New function `IsCoercible` to test whether an object is coercible into a given structure, and the result of coercion if allowed. The effect is similar to `!` but allows a test for failure.

– New function `MakeType` to create the type (category name) named by a given string.

– New function `IsIntrinsic` which tests whether a specific intrinsic exists, named by a given string, and if so, returns the actual intrinsic.

– New function `GetVersion` which returns a sequence of integers indicating the current version of Magma.

– New procedure `SetTraceback` and function `GetTraceback` to control whether Magma should produce a traceback of user function calls before each error message.

– New procedure `Traceback` to display a traceback of the current Magma function invocations.

– Function `HasOutputFile` is documented for the first time.

– Function `Realtime` is documented for the first time.

– Procedure `ShowMemoryUsage` is documented for the first time.

5 Aggregate Data Types [HB 6–9]

New features:

– New function `CanChangeUniverse`, to test whether the universe of a set or sequence can be changed (like `ChangeUniverse` but with a test for success).

– New functions and procedures (`Append(T, x)`, `Append(~T, x)`, `Prune(T)` and `Prune(~T)`) for appending to and pruning of tuples [HB 9].

6 Groups and Semigroups

6.1 General Groups [HB 15]

Removals and Changes:

– The library `gps100` has been withdrawn. It is subsumed in the much extended database `SmallGroups` (see below).

New features:

– A database of all groups having order at most 1000 (excepting orders 512 and 768) has been included in Magma. This database was prepared by Bettina Eick and Hans Ulrich Besche in 1997 using GAP, and incorporates directly the libraries of 2-groups of order dividing 256 and the 3-groups of order dividing 729 which were prepared by M.F. Newman and E.A. O'Brien. The functions available for accessing the database are

  – `SmallGroupDatabaseLimit`,
  – `NumberOfSmallGroups(`n`)`,
  – `SmallGroup(`o, n`)`,
  – `SmallGroups(`o`)`, and
  – `SmallGroupProcess(`o`)`.

  Functions `IsEmpty`, `Current`, `CurrentLabel` and `Advance` are provided for working with a small group process.

– The function `CharacterDegrees` employs Slattery's algorithm to determine the degrees of the irreducible characters of a p-group.

6.2 Finitely Presented Groups [HB 16]

New features:

– Given a finitely presented group G and normal subgroups H and K of G such that $K \leq H \leq G$, and a prime p, the function `GModule(`G, H, K, p`)` returns the $K[G]$-module M corresponding to the action of G on the largest elementary abelian factor of H/K of p-power order. If P is omitted, the module returned is the largest elementary abelian factor of H/K.

– Given a finitely presented group G and normal subgroup H of G and a prime p, the function `Gmodule(`G, H, p`)` returns the $K[G]$-module M corresponding to the action of G on the largest elementary abelian factor of H of p-power order.

6.3 Finitely Generated Abelian Groups [HB 18]

New features:

– Given finite abelian groups G and H, the function `Hom(`G, H`)` computes an abstract abelian group isomorphic to $E = \mathrm{Hom}(G, H)$. In addition, a transfer map is returned which converts an element of E to the actual homomorphism. Related functions are `HomGenerators` and `AllHomomorphisms`.

6.4 Finite Soluble Groups [HB 19]

New features:

- If the abelian groups G and H are given in terms of power-conjugate presentations, the function `Hom(`G, H`)` computes an abstract abelian group isomorphic to $E = \mathrm{Hom}(G, H)$. In addition, a transfer map is returned which converts an element of E to the actual homomorphism. Related functions are `HomGenerators` and `AllHomomorphisms`.
- The function `CharacterDegrees` employs Slattery's algorithm to determine the degrees of the irreducible characters of a p-group.

6.5 Permutation Groups [HB 20]

Removals and Changes:

- The library `prmgps` of primitive groups has been withdrawn. It is replaced by the database `PrimitiveGroups` (see below).
- The library `trngps` of transitive groups has been withdrawn. It is replaced by the database `TransitiveGroups` (see below).

New features:

- The various functions that compute conjugacy classes of subgroups of a permutation group G have been modified to compute the conjugacy classes of subgroups of a quotient group of G. Thus, if N is a normal subgroup of G, the function `Subgroups(`G, N`)` will compute a preimage in G of a representative for each conjugacy class of subgroups of the quotient group G/N. The optional parameters that apply to `Subgroups(`G`)` also apply to this variant.
- The primitive groups library `prmgps` has been converted into a database. Thus, the primitive groups of degree up to 50 are now directly accessible from within Magma (without having to load the library), and the names of the access functions have been redesigned. The new functions are:
  - `PrimitiveGroupDatabaseLimit`,
  - `NumberOfPrimitiveGroups` (replacing `PrmNumberOfDegree`),
  - `PrimitiveGroup` (replacing `PrmGroup`, `PrmGroupSatisfying`, `PrmGroupOfDegreeSatisfying`),
  - `PrimitiveGroups` (replacing `PrmGroupsSatisfying`, `PrmGroupsOfDegreeSatisfying`),
  - `PrimitiveGroupDescription` (replacing `PrmInfo`), and
  - `PrimitiveGroupProcess` (replacing `PrmProcess`, `PrmProcessOfDegree`).

  The functions `IsEmpty`, `Current`, `CurrentLabel` and `Advance` are provided for working with a primitive group process, replacing `PrmProcessIsEmpty`, `PrmProcessGroup`, `PrmProcessLabel` and `PrmProcessNext`.
- The transitive groups library `trngps` has been converted into a database. Thus, the Hulpke tables of transitive groups of degree up to 22 are now directly accessible from within Magma (without having to load a library), and the names of the access functions have been redesigned. The new functions are:
  - `TransitiveGroupDatabaseLimit`,

- `NumberOfTransitiveGroups` (replacing `TrnNumberOfDegree`),
- `TransitiveGroup` (replacing `TrnGroup`, `TrnGroupSatisfying`, `TrnGroupOfDegreeSatisfying`),
- `TransitiveGroups` (replacing `TrnGroupsSatisfying`, `TrnGroupsOfDegreeSatisfying`), and
- `TransitiveGroupProcess` (replacing `TrnProcess`, `TrnProcessOfDegree`).

The functions `IsEmpty`, `Current`, `CurrentLabel` and `Advance` are provided for working with a transitive group process, replacing `TrnProcessIsEmpty`, `TrnProcessGroup`, `TrnProcessLabel` and `TrnProcessNext`. A new function `TransitiveGroupName` has been installed; it returns a string giving the name of the group in the Hulpke database.

- The function `TransitiveGroupIdentification(`G`)` returns the index in the transitive group database of the group that is isomorphic to a given transitive group G.
- A recursive algorithm due to Bill Kantor for computing the maximal normal p-subgroup of a permutation group is now available as the function `pCoreKantor`. Related functions are `MEANS`, `ElementaryAbelianNormalSubgroup` and `pElementaryAbelianNormalSubgroup`.

Bug fixes:

- A bad memory management technique for the partition-backtrack permutation group algorithms which caused very great loss of performance for large-memory runs has been fixed.
- A bug in the implementation of a new algorithm for computing normal subgroups of permutation groups (first installed in V2.3) has been fixed. The effect of this bug was sometimes to cause a crash and sometimes to result in some subgroups being missed. (Reported by Jurgen Klüners.)
- A bug that sometimes caused incorrect results to be returned when computing the conjugacy classes of elements of a soluble permutation group has been fixed. (Reported by Gregor Kemper.)

6.6 General Matrix Groups [HB 21]

New features:

- New function `VectorSpace(G)` to return the vector space on which the matrix group G acts naturally (assuming G is over a field; otherwise the same as the function `RSpace`).
- The function `Stabilizer` now works when given an element (vector) of a G-module and an appropriate matrix group.
- New function `IntegralGroup(`G`)` which returns a group of integer matrices conjugate to the rational matrix group G.

6.7 Matrix Groups over Finite Fields [HB 21]

The basic facilities provided by Magma for computing with matrix groups over finite fields depend upon being able to construct a chain of stabilizers. However, there are many examples of groups of moderately small degree where we cannot find a suitable chain.

An on-going international research project seeks to develop algorithms to explore the structure of such groups. The main theoretical underpinning of the project comes from the classification by Aschbacher (1984) of the (maximal) subgroups of $\mathrm{GL}(d, q)$ into nine

families. Much of the research effort to date has been devoted to designing algorithms to decide whether G belongs to one of the eight families whose members have a normal subgroup preserving a "natural linear structure"; here, we plan to exploit this information to explore G further, ultimately producing a composition series for G. The final family consists of groups G which are almost simple modulo scalars; here the aim is to determine, a pair of efficiently computable inverse isomorphisms between G and a standard copy of the relevant simple group.

New features:

- A family of functions is provided for testing whether a group preserves a form modulo scalars. The main functions are `ClassicalForms`, `SymmetricBilinearForm`, `QuadraticForm`, and `UnitaryForm`. Corresponding functions are provided to return the scalars corresponding to the generators of the group of the form. These functions were implemented in Magma by Alice Niemeyer and Anthony Pye.
- A family of functions is provided for testing whether a group is a classical group in its natural representation. The main functions are `RecognizeClassical`, `IsLinearGroup`, `IsSymplecticGroup`, `IsOrthogonalGroup` and `IsUnitaryGroup`. These functions apply the Niemeyer-Prager classical group recognition algorithm as implemented in Magma by Alice Niemeyer and Anthony Pye.
- The function `IsPrimitive` determines whether a subgroup G of $\mathrm{GL}(d, q)$ acts imprimitively on the underlying vector space. The ancillary functions `BlockSystem` and `BlocksImage` return information about a proper block system and the action of G on such a block system, respectively. This group of functions was implemented in Magma by Eamonn O'Brien.
- The function `IsSemiLinear` tests whether a matrix group G acts as a semilinear group of automorphisms on some vector space. If G does act semilinearly, information about this action is provided by the functions `DegreeOfFieldExtension`, `CentralisingMatrix`, `FrobeniusAutomorphisms`, and `WriteOverLargerField`. This group of functions was implemented in Magma by Eamonn O'Brien.
- The function `IsTensor` tests whether a matrix group G preserves a non-trivial tensor product decomposition. If G does so, information about the decomposition may be obtained using the functions `TensorBasis`, `TensorFactors`, and `IsProportional`. This group of functions was implemented in Magma by Eamonn O'Brien.
- The function `SearchForDecomposition` searches for certain types of decompositions (corresponding to some of the Aschbacher families) of a matrix group with respect to the normal closure of a supplied subgroup. The following additional functions `IsExtraSpecialNormaliser`, `ExtraSpecialParameters`, `ExtraSpecialGroup`, `IsSymmetricTensor`, `SymmetricTensorBasis`, `SymmetricTensorFactors`, `SymmetricTensorPermutations` provide access to information about some of the decompositions. This group of functions was implemented in Magma by Eamonn O'Brien.
- The function `IsOverSmallerField` uses the Glasby-Howlett algorithm to decide if the absolutely irreducible group $G \leq \mathrm{GL}(d, K)$ has an equivalent representation over a subfield of K.
- Given a group G of $d \times d$ matrices over a finite field E having degree e and a subfield F of E having degree f, the function `WriteOverSmallerField` returns a group generated by the matrices of G written as $de/f \times de/f$ matrices over F.

7 Rings

7.1 General Rings [HB 23]

New features:

- New functions `GCD` and `LCM` taking a set or sequence of general ring elements.
- The function `IsDivisibleBy` is now available for many ring element types for the first time.
- New function `ExactQuotient` for exact quotient (see `IsDivisibleBy`).

7.2 Integer Ring [HB 24]

The Schönhage-Strassen FFT-based algorithm for the multiplication of very large integers has been implemented. The crossover point (when this method beats the Karatsuba method) is currently 2^{15} bits (approx. 10000 decimal digits) on Sun Sparc workstations and 2^{17} bits (approx. 40000 decimal digits) on Digital Alpha workstations. The product of two arbitrary integers, each having one million *decimal* digits, may now be computed in 5.1 seconds on a 250MHz Sun Ultrasparc, or in 2.0 seconds on a 333MHz Digital Alpha workstation (a 64-bit machine).

Magma V2.4 also contains an implementation of an asymptotically-fast integer (and polynomial) division algorithm which reduces division to multiplication with a constant scale factor that is in the practical range. Thus division of integers and polynomials are now based on the Karatsuba and Schönhage-Strassen (FFT) methods when applicable. The crossover point for integer division (when the new method outperforms the classical method) is currently at the point of dividing a 2^{12} bit (approx. 1200 decimal digit) integer by a 2^{11} (approx. 600 decimal digit) integer on Sun Sparc workstations.

Changes:

- The function `IsDivisible` has been renamed to `IsDivisibleBy`.
- The function `SquareFree` has been renamed to `Squarefree` and `SquareFreeFactorization` has been renamed to `SquarefreeFactorization`; the old function names have been removed from the documentation but kept in the executable to allow old code to continue to work for this release.
- The function `ChineseRemainderTheorem` or `CRT` taking three integers has been removed to avoid confusion. (`Solution` still retains the original functionality.)
- The function `ChineseRemainderTheorem` or `CRT` taking three integer sequences has been changed so that it now only takes two integer sequences (the original "multipliers" sequence is now considered to consist of all ones). (`Solution` still retains the original functionality.)

Summary of new features:

- New Schönhage-Strassen FFT method for fast multiplication of very large integers.
- New asymptotically-fast division algorithm.
- The function `IsSquare` now works for any integer.
- New function `Ilog` to return the floor of the logarithm of an integer to a given integer base.

7.3 Finite Fields [HB 27]

Magma V2.4 contains a new packed representation for polynomials over GF(2) which is used to represent elements of finite fields of characteristic 2. This has led to dramatic speed-ups for computations within such fields. Magma V2.3 used a two-step representation where a large field was represented as an extension field of a Zech representation field, but this representation could only be used if the degree of the field had a suitable divisor and was not too large. The prime factorization of the degree of the field is irrelevant in the new representation so, for example, computation in fields of characteristic 2 with large prime degree is now very much faster.

A database of sparse irreducible polynomials over GF(2) has been constructed for all degrees up to 11000. Thus any finite field $\mathrm{GF}(2^k)$, for k within this range, may be created without any delay. The sparseness of the defining polynomial is utilized in the arithmetic. Computation in fields such as $\mathrm{GF}(2^{10000})$ and $\mathrm{GF}(2^{10007})$ (10007 is prime) is now quite feasible.

The Shoup algorithm for factoring polynomials over large finite fields has been implemented in V2.4. Combined with the great improvements to the finite field arithmetic itself, factorization over large finite fields is now very much faster in V2.4 than in previous versions. For example, factorizing a polynomial over $\mathrm{GF}(2^{1000})$ is about 500–1000 [sic] times faster in V2.4 than in V2.3. For fields of higher degree, the improvement is even greater.

On a 64-bit 200MHz SGI Origin 2000, the $n = 2048$ polynomial from the von zur Gathen challenge benchmarks (of degree 2048 with sparse coefficients modulo a 2050-bit prime) is factored by Magma V2.4 in about 18.8 hours and the $n = 3000$ polynomial from the same benchmarks (of degree 3000 with sparse coefficients modulo a 3002-bit prime) is factored by Magma V2.4 in about 75.8 hours. Also, the $n = 2048$ polynomial from the Shoup challenge benchmarks (of degree 2048 with dense coefficients modulo a 2048-bit prime) is factored by Magma V2.4 in about 36.0 hours on the same machine. (The times taken for the same problems are about twice as long on a 250MHz Sun Ultrasparc.)

Summary of new features:

- New database of sparse irreducible polynomials over GF(2) for all degrees up to 11000 (accessed by `IrreduciblePolynomial(GF(2), d)` and also used internally by functions such as the creation function `FiniteField`).
- The arithmetic for univariate polynomials over finite fields, and arithmetic of medium-sized non-prime fields has been sped up by use of FFT methods, etc. (see Univariate Polynomial Rings below).
- Using a new efficient packed representation, arithmetic with finite fields of characteristic 2 has been dramatically improved.
- New Shoup factorization algorithm for polynomials over finite fields (automatically selected by `Factorization` when appropriate or by a parameter).
- The computation of square roots of elements of finite fields of characteristic 2 has been sped up.
- The function `FiniteField`(p, n) now has a parameter `Check`; setting `Check` to `false` will cause the function to skip the check that p is prime.
- New function `NormEquation` to solve norm equations in finite fields.

– New function `AllRoots` to return all n-th roots of a given element of a finite field.

– New function `RootsInSplittingField(f)`, which, given a polynomial over a finite field K, computes a splitting field S of f as an extension field of K, and returns the roots of f in S, together with S.

– New function `FactorizationOverSplittingField(f)`, which, given a polynomial over a finite field K, computes a splitting field S of f as an extension field of K, and returns the factorization (into linears) of f over S, together with S.

– New function `Log(b, x)` to compute the discrete logarithm of x to the base b, for any primitive element b.

– New procedure `SetPrimitiveElement(K, x)` to set the internal primitive element of K to be x (useful when computing many logarithms with respect to a particular base).

– The computation of discrete logarithms has been sped up (particularly for fields whose cardinality is less than 10^{10}). The function `Log` may now be applied to an element of any finite field of arbitrary size. The Pohlig-Hellman algorithm for computing discrete algorithms is now combined with both the Shanks baby-step/giant-step and Pollard-ρ algorithms. In V2.4, computing the logarithm of any element of a field having cardinality less than 10^{10} takes less than a second, and computing the logarithm of any element of a field where the largest prime divisor of the order of the multiplicative group is 15 decimal digits now takes about 100 seconds (on a 250MHz Sun Ultrasparc).

7.4 Univariate Polynomial Rings [HB 28]

Magma V2.4 incorporates new fast methods for performing arithmetic with univariate polynomials. These include two FFT-based methods for multiplication: the Schönhage-Strassen FFT method where the coefficients are large compared with the degree, and the small-prime modular FFT with Chinese remaindering method where the coefficients are small compared with the degree. These new methods are applied to multiplication of polynomials over the integer ring, the rational field, integer residue class rings and prime finite fields. For some kinds of coefficients, at least one of the FFT methods beats the Karatsuba method as low as degree 32 or 64; for all of the kinds of coefficients listed, the FFT methods beat the Karatsuba method for degree 128 or greater.

An asymptotically-fast division algorithm (which reduces division to multiplication) is now used for polynomials over all coefficient rings (similar to the algorithm used for integers).

Factorization of univariate polynomials over all finite field types has been dramatically improved—see the section on Finite Fields above.

Changes:

– The function `SquareFreeFactorization` has been renamed to `SquarefreeFactorization`; the old function name has been removed from the documentation but kept in the executable to allow old code to continue to work for this release.

Summary of new features:

– New Schönhage-Strassen FFT algorithm for multiplication of polynomials (with large coefficients).

– New small-prime modular FFT algorithm for multiplication of polynomials (with small coefficients).

– New asymptotically-fast division algorithm.

– Modular exponentiation sped up by use of pre-inversion of modulus and fast multiplication (and thus leading to a speed up in all algorithms, such as factorization, which use this).

– New fast modular algorithm for the computation of the extended GCD of two univariate polynomials over the rational field.

– Factorization of polynomials over all finite fields greatly improved—see the section on Finite Fields.

– New parameter `Global` for polynomial ring creation functions. This now allows the creation of multiple non-global polynomial rings with different generator names.

– The function `Evaluate` for univariate polynomial rings has been changed and extended to use the common over-ring of its arguments; this overcomes some previous confusing behaviour caused by automatic coercion.

– The common over-structure of univariate polynomial rings is now supported for the first time (used by automatic coercion).

– New function `HasPolynomialFactorization(R)` to determine whether factorization of polynomials over the ring R is allowed in Magma.

– New function `Valuation` for univariate polynomials.

7.5 Multivariate Polynomial Rings [HB 29]

New algorithms for factorization of bivariate polynomials and general multivariate polynomials over finite fields have been implemented. The previous implementations failed on various inputs and were rather unsatisfactory.

Changes:

– The function `Variety` now returns a sequence of tuples.

– The function `SquareFreeFactorization` has been renamed to `SquarefreeFactorization`.

New features:

– The dimension of a full polynomial ring (considered as an ideal) is now defined to be -1, so the functions `Dimension`, `IsZeroDimensional`, etc. may be applied to such.

– New improved algorithm for factorization of multivariate polynomials over finite fields.

– New fast algorithm for the computation of GCDs of multivariate polynomials over all fields of characteristic zero using evaluation-interpolation techniques.

– New fast algorithm for factorization of multivariate polynomials over algebraic number fields (including quadratic and cyclotomic fields) by better use of resultants.

– New function `TriangularDecomposition` to compute the triangular decomposition of zero-dimensional ideals (algorithm of D. Lazard).

– The primary decomposition algorithm has been significantly improved by use of the triangular decomposition algorithm together with other techniques.

7.6 Invariant Rings of Finite Groups [HB 30]

New features:

- The functions `IsInvariant(f, g)` and `IsInvariant(f, G)` have been documented for the first time.
- The functions `PrimaryAlgebra` and `PrimaryIdeal` have been documented for the first time.

7.7 Algebraic Number Fields [HB 35]

The KANT number field machinery corresponding to Kash 1.9 has been installed. This is the first major upgrade of the Magma general number field facility since Kash V1.6 was installed in early 1996. This version of KANT provides the Magma user with greatly enhanced performance for many fundamental algorithms. Of particular note is the new KANT maximal order algorithm that combines both the Round 2 and Round 4 methods. Both conditional and unconditional computation of class groups has been greatly improved and the result is far superior performance to that provided in previous versions of Magma.

Major new number field facilities available in Magma V2.4 (courtesy of KANT) are ray class groups, S-unit groups, prime ideal decompositions of fractional ideals, automorphism groups of normal and abelian extensions, solution of Thue equations, unit equations and index form equations.

Changes:

- The function `BetterPolynomial` has been replaced by `OptimizedRepresentation`. The function `BetterPolynomial` will remain for a transition period.
- The function `MaximalOrder` has new parameters reflecting changes to the underlying algorithm.
- The function `ClassGroup` has new parameters reflecting changes to the underlying algorithm.
- The number field function `BetterPolynomial` has been replaced by `OptimizedRepresentation` which has slightly different semantics. The old function will be retained for compatibility in the medium term.
- All number field functions that return (inexact) real or complex numbers have been changed so that instead of returning fixed precision numbers, they now return free numbers (type FldPrElt).

New features:

- The function `RadicalExtension` allows easy construction of radical extensions.
- The function `CompositeFields` creates the composition of two number fields.
- The function `SplittingField` will construct the splitting field of a number field directly.
- The function `RelativeField` allows the construction of relative extensions.
- The function `AbsoluteDegree` gives the absolute degree of an order or number field.
- The functions `IsNormal` and `IsAbelian` test a number field for being a normal (respectively normal abelian) extension of $\mathbf{Q}$.

– The functions `CharacteristicPolynomial` and `AbsoluteCharacteristicPolynomial` give the characteristic polynomial of a field element.

– The function `Divisors`, applied to an algebraic integer, returns all its divisors.

– The function `Zeros` gives the zeros of the defining polynomial of a field or order.

– The functions `Index` gives the index of an ideal in an over-order.

– The infix operator `meet` has been extended so as to compute the intersections of two ideals belonging to an order.

– The function `Decomposition`, applied to a rational prime p, finds the factorization of p into prime ideals in a designated order. If this function is applied to an order element a it finds the factorization of a into prime ideals.

– The function `InertiaDegree` gives the degree of inertia of a prime ideal.

– The function `Verify` is provided for checking a conditional class group computation.

– Unconditional and conditional (GRH) computation of ray class groups is provided by the function `RayClassGroup`.

– The function `ClassRepresentative`, applied to an ideal I belonging to an order whose class group is known, will give the chosen representative of the ideal class containing I.

– The group of S-units corresponding to a set S of prime ideals is provided by the function `SUnitGroup`.

– Function `TorsionUnitGroup` gives the torsion part of the unit group of a number field or order.

– The machinery developed by Klüners for computing subfields has been greatly improved. It has been sucessfully used to compute all subfields in an extension of degree 60.

– The functions `Automorphisms` and `AutomorphismGroup` provide for the calculation of the automorphism group of a normal extension.

– The machinery for solving Thue equations has been improved and an additional function `ThueEval`, for evaluating the homogeneous form, is provided.

Bug fixes:

– A number of bugs present in V2.3 and earlier versions, and arising in the computation of class groups, unit groups and testing ideals for being principal are fixed with the installation of the new version of KANT.

7.8 Rational Function Fields, [HB 35]

New features:

– The function `Evaluate` for univariate function fields has been changed and extended to use the common over-ring of its arguments; this avoids its previous confusing behaviour caused by automatic coercion.

– New function `Derivative` for function field elements (4 separate signatures).

– New function `PartialFractionDecomposition` which computes the complete partial fraction decomposition of a function field element. (`PartialFractionDecomposition` now factorizes fully and the original functionality is now given by the function `SquarefreePartialFractionDecomposition`.)

– Coercion from one rational function field to a compatible field is now allowed and the common over-structure of two function fields is computed correctly.

– New functions `Degree`, `TotalDegree` and `WeightedDegree` for function field elements.

7.9 Real and Complex Fields [HB 36]

New features:

– The function `ExponentialIntegralE1` takes a real number r and calculates the principal value of

$$\int_x^\infty \frac{e^u}{u} du$$

at $x = r$.

Bug fixes:

– A bug in Xavier Gourdon's root finding program in the case of badly conditioned polynomials has been fixed.

7.10 Formal Series [HB 37]

The module for computing with formal series has been completely rewritten for Magma V2.4. Many bugs and leaks have been removed in the process.

The arithmetic for series over the rational field is generally 5 times faster (as a consequence of the use of fraction-free methods). Asymptotically-fast methods for arithmetic are now used for the first time (based on the new integer and polynomial methods). The 10000-th Bernoulli number B_{10000} may be computed in about 14 hours on a 250MHz Sun Ultrasparc directly from its generating function.

New features:

– Series with fractional exponents are now available. Arbitrary exponent denominators are allowed and such series may be freely mixed. Practically all series functions apply to series with fractional exponents if the result is meaningful and computable.

– The new category names for series rings and series are: `RngSerPow` (power series ring), `RngSerPowElt` (power series), `RngSerLaur` (Laurent series ring), `RngSerLaurElt` (Laurent series), `RngSerPuis` (Puiseux series ring), `RngSerPuisElt` (Puiseux series), `RngSerPuis` (any series ring), `RngSerPuisElt` (any series).

– The functions `PowerSeriesAlgebra` and `LaurentSeriesAlgebra` have been removed; the new creation functions are `PowerSeriesRing`, `LaurentSeriesRing` and `PuiseuxSeriesRing`.

– New parameter `Global` for series ring creation functions. This now allows the creation of multiple non-global series rings with different generator names.

– New function `ExponentDenominator` to return the denominator of exponents of a Puiseux series.

– New functions `LeadingCoefficient` and `LeadingTerm`.

– The function `Reversion` now works (for the first time) in the case of fields having non-zero characteristic and for series with general positive valuation.

– Intrinsics for the inverse trigonometric functions `Arcsin`, `Arccos` and `Arctan` and inverse hyperbolic functions `Argsinh`, `Argcosh`, `Argtanh` have been installed.

– Intrinsics for the trigonometric functions `Sec`, `Cosec` and `Cot` have been installed.

– The intrinsic function `HypergeometricSeries` is provided for computing hypergeometric series.

– New functions `AGM`, `Gamma`, `LogGamma` and `Polylog`.

– New intrinsic functions for computing with Elliptic and Modular functions have been implemented: Jacobi θ (`JacobiTheta`), Dedekind η (`DedekindEta`), j-invariant (`jInvariant`), Δ function (`Delta`), Eisenstein function (`Eisenstein`) and Weierstrass $\wp$-function (`WeierstrassSeries`).

8 Modules

8.1 Vector Spaces and R-spaces [HB 40]

New features:

– New function `Moduli` to return the column moduli of an R-space over a Euclidean domain.

8.2 R-Modules [HB 41]

New features:

– Function `SingleMinimalSubmodule` documented for the first time.

9 Algebras

9.1 Group Algebras [HB 49]

Changes:

– The `sub`-constructor for group algebras now returns a sub-group algebra (in category `AlgGrpSub`) instead of an associative algebra. This is more natural.

– Ideals and subalgebras now make use of generators resulting in a very considerable speed-up in closure operations.

9.2 Matrix Algebras [HB 50]

New features:

– Genuine 64-bit versions for packed matrices over GF(2) have been implemented for 64-bit machines.

– The function `Adjoint` now works over any ring which has an exact division algorithm (and is now correct for the first time!).

– A matrix algebra element over a ring R is now coercible into a matrix group over another ring S if elements of R are coercible into S.

10 Geometry

10.1 Elliptic Curves [HB 52]

The facilities for elliptic curves have undergone major expansion. Noteworthy is the introduction of general machinery for working with isomorphisms, isogenies and rational maps between curves. The implementation introduces new data types for such things as subgroups and subschemes of elliptic curves. The image of a subgroup under an isogeny (whose kernel is contained in the subgroup) can be calculated, an operation which is used to construct an isogeny from its kernel but is also of independent interest.

The implemented functions together form a powerful toolkit for working with elliptic curves. The interplay between elliptic curves, isogenies and subgroups is particularly beneficial. The toolkit allows comparatively easy computation of structures such as the endomorphism ring of a curve over a finite field and the eigenvalues of the Frobenius automorphism (the latter being useful in point-counting).

Changes:

- The category name for elliptic curves has been changed to `CurveEll` and the category name for elliptic curve points has been changed to `CurveEllPt`.

10.1.1 General Elliptic Curves

New features:

- Extension and lifting of curves induced by maps of base rings: `BaseExtend`, `ChangeRing`.
- Division polynomials and m-torsion subgroups: `DivisionPolynomial`. `mTorsionSubgroup`.
- The (starred) modular equations are currently available for all primes in the range 23 to 293: `ModularEquation`.
- Order of a point an elliptic curve: `Order`.
- The ζ-function of an elliptic curve at a given prime: `ZetaFunction`.
- Creation of isogenies, isomorphisms, rational maps between curves and translation maps on a curve: `Morphism`, `Isogeny`, `Automorphism`, `TranslationMap` and `RationalMap`.
- Velu's formula for constructing an isogeny with a given kernel: `IsogenyFromKernel`, and `IsogenyFromKernelFactored`.
- Polynomials defining isogenies: `IsogenyMapOmega`, `IsogenyMapPhi`, `IsogenyMapPhiMulti`, `IsogenyMapPsi`.
- Kernel and image of an isogeny as subgroups, image and preimage of a subgroup under an isogeny: `Kernel`, `PushThroughIsogeny`, etc.
- Operations with isogenies: degree, composition, construction with given kernel, Frobenius endomorphism: `Degree`, etc.
- Operations with isomorphisms: inverses, composition.
- Test whether two elliptic curves are isogeneous or isomorphic: `IsIsogenous` and `IsIsomorphism`.

– Subgroups of an elliptic curve as a type: `Subgroup`, `Order`, `mTorsionSubgroup` and `RationalPoints`.
– Subschemes of an elliptic curve as a type: `Subscheme`, `DefiningIdeal`, and `RationalPoints`.
– The database of elliptic curves having conductor up to 5300 constructed by John Cremona is now included: `CremonaDatabase` and many others.

10.1.2 Elliptic Curves over Finite Fields

In V2.4 the computation of the order of an elliptic curve over a finite field is performed using the plain Schoof algorithm (with the restriction that the characteristic of the field must be greater than 3). An efficient implementation of the Schoof-Elkies-Atkin-Lercier algorithm is under development and will be available in Magma V2.5.

New features for elliptic curves over finite fields:

– Plain Schoof algorithm for counting the points on a curve over a field having characteristic greater than 3 `Order`.
– Trace of a curve (function `Trace`).
– Random point of a curve (function `Random`).
– Quadratic twist of a curve (function `QuadraticTwist`).
– Non-deterministic tests for deciding whether a curve is supersingular: `IsProvenSupersingular`, `IsProbablyOrdinary`.
– Enumeration of all points of a curve (for small fields).
– Function `ModularEquation` to return Atkin's variation of the modular polynomial for a given prime p.

11 Incidence Structures

11.1 Enumerative Combinatorics [HB 53]

Changes:

– The function `BernoulliNumber` has been sped up very significantly by the use of power series (with asymptotically-fast arithmetic).

11.2 Incidence Structures and Designs [HB 55]

Changes:

– Most point and block functions now take an extra argument: the incidence structure in which the operation should be performed. This allows for more flexibility when working with incidence structures, and makes the incidence structure module more consistent with others such as groups.

New features:

– New function `HadamardNormalize` to normalize a Hadamard matrix to have only ones in the first row and first column.
– New function `HadamardAutomorphismGroup` to compute the automorphism group of a Hadamard matrix.

11.3 Finite Planes [HB 56]

Facilities for subplanes of finite projective and affine planes have been improved so as to make working with planes and subplanes much easier.

Changes:

- Most point and line functions now take an extra argument: the incidence structure in which the operation should be performed. This allows for more flexibility when working with planes and makes the module compatible with others.
- The following plane intrinsics have changed to accept the plane in which the operation should be performed as the first argument: `AllPassants`, `AllSecants`, `AllTangents`, `CentralCollineationGroup`, `Conic`, `ContainsQuadrangle`, `Coordinates`, `Exterior`, `ExternalLines`, `Index`, `Interior`, `IsArc`, `IsCollinear`, `IsComplete`, `IsConcurrent`, `IsParallel`, `IsUnital`, `Knot`, `ParallelClass`, `Pencil`, `QuadraticForm`, `Support`, `Tangent`, `UnitalFeet`.

11.4 Error-correcting Codes [HB 57]

New features:

- The function `IsEquivalent` has been documented for the first time.