MODULAR ABELIAN VARIETIES by William Stein

0.1 Introduction

This chapter is the reference for the modular abelian varieties package in MAGMA. A *modular abelian variety* is an abelian variety that is a quotient of the modular Jacobian $J_1(N)$, for some integer N. This package provides extensive functionality for computing with such abelian varities. This includes functions for enumerating and decomposing modular abelian varieties, isomorphism testing, computing exact endomorphism and homomorphism rings, doing arithmetic with finite subgroups, and computing information about torsion subgroups, special values of L-functions, and Tamagawa numbers.

Essentially none of the algorithms in this package use explicit defining equations for varieties, and as such that work in a great degree of generality. For example, many even make sense for Grothendieck motives attached to modular forms, and we have included the corresponding functionality here, when it makes sense.

This is the first release of the modular abelian varieties package, so it could probably be optimized more. The major drawback of the current version is that complete decomposition into simples is only currently implemented over the rational numbers. Thus the interesting behavior over number fields, involving extra inner twists, which leads to $\mathbf{Q}$-curves and associated questions, is not available (much of it is implemented, but there are some fundamental theoretical obstructions I am working on overcoming).

Our philosophy for representing modular abelian varieties is perhaps different than what you might expect, so we describe how we view abelian subvariety A over $\mathbf{Q}$ contained in the modular Jacobian $J_0(N)$. By the Abel-Jacobi theory we may view $J_0(N)$ over the complex numbers as a complex vector space V modulo the lattice $H_1(J_0(N), \mathbf{Z}) = H_1(X_0(N), \mathbf{Z})$. An abelian subvariety $A \subset J_0(N)$ and the map $i : A \to J_0(N)$ is completely determined by giving the image of $H_1(A, \mathbf{Q})$ in the vector space $H_1(X_0(N), \mathbf{Q})$. At this point, it might appear that we have to compute lots of floating point numbers and approximate lattices in the complex numbers, but this is not the case. Instead, we use modular symbols to compute $H_1(X_0(N), \mathbf{Z})$ as an abstract abelian group, and use everything we can from the extensive theory of modular forms to compute things about the abelian varieties determined by subgroups of $H_1(X_0(N), \mathbf{Z})$, and other related abelian varieties. Note that even though we work with homology, which is associated to complex tori, the abelian variety A over $\mathbf{Q}$ is still determined by our defining data (a certain subgroup of $H_1(X_0(N), \mathbf{Z})$), and our algorithms can often take advantage of this.

0.1.1 Categories

Modular abelian varieties belong to the category `ModAbVar`, and the elements of modular abelian varieties belong to `ModAbVarElt`. The category `MapModAbVar` consists of homomorphisms between modular abelian varieties (sometimes only up to isogeny, i.e., with a denominator). Finitely generated subgroups of modular abelian varieties form the category `ModAbVarSubGrp`. Spaces of homomorphisms between modular abelian varieties form the category `HomModAbVar`. Homology of a modular abelian variety is in the category `ModAbVarHomol`. The L-series of modular abelian varieties are in `ModAbVarLSer`.

Example H0E1

We create an object of each category.

```
> A := Jzero(11);
> Type(A);
ModAbVar
> Type(A!0);
ModAbVarElt
> Type(nIsogeny(A,2));
MapModAbVar
> Type(nTorsionSubgroup(A,2));
ModAbVarSubGrp
> Type(End(A));
HomModAbVar
> Type(Homology(A));
ModAbVarHomol
> Type(LSeries(A));
ModAbVarLSer
```

0.1.2 Verbose Output

To set the verbosity level use the command `SetVerbose("ModAbVar",n)`, where `n` is 0 (silent), 1 (verbose), or 2 (very verbose). The default verbose level is 0.

Two new additional verbose levels are included in this package. Level 3 is exactly like level 1, except instead of displaying to the screen, verbose output is appended to the file `ModAbVar-verbose.log` in the directory that MAGMA was run from. Verbose level 4 is exactly like level 2, except verbose output is appended to `ModAbVar-verbose.log`. On a UNIX-like system, use the shell command `tail -f ModAbVar-verbose.log` to watch the verbose log in another terminal.

Example H0E2

Using `SetVerbose`, we get some information about what is happening during computations.

```
> SetVerbose("ModAbVar",1);   // some verbose output
> SetVerbose("ModAbVar",2);   // tons of verbose output
> SetVerbose("ModAbVar",3);   // some verbose output to ModAbVar-verbose.log
> SetVerbose("ModAbVar",4);   // tons of verbose output to ModAbVar-verbose.log
```

0.2 Creation and Basic Functions

The functions described below are for creating modular abelian abelian varieties, combining them together in various ways, and obtaining simple information about them.

Modular abelian varieties are much less restricted than spaces of modular symbols, in that one can take arbitrary finite direct sum.

0.2.1 Creating the Modular Jacobian J0(N)

Use the `Jzero` command to create the Jacobian $J_0(N)$ of the modular curve $X_0(N)$ (which parameterizes pairs consisting of an elliptic curve and a cyclic subgroup of order N). You can also create higher weight motivic analogues of this Jacobian, and you can compute in the $+1$ or -1 quotient of homology for efficiency, though certain results will be off by factors of 2.

`Jzero(N : parameters)`

`sign` RngIntElt *Default :* 0

Create the modular abelian variety $J_0(N)$, i.e., the Jacobian of the modular curve $X0(N)$.

`Jzero(N, k : parameters)`

`sign` RngIntElt *Default :* 0

Create the modular abelian variety $J_0(N)$ of weight $k \geq 2$.

`Jzero(N, k, sign)`

Create the modular abelian variety $J_0(N)$ of weight $k \geq 2$.

Example H0E3

```
> Jzero(23);
Modular abelian variety Jzero(23) of dimension 2 and level 23 over Q
> Jzero(23 :  sign := +1);
Modular abelian variety Jzero(23) of dimension 2 and level 23 over Q
with sign 1
> Jzero(23,4);
```

```
Modular motive Jzero(23,4) of dimension 5 and level 23 over Q
> Jzero(23,4 :  sign := -1);
Modular motive Jzero(23,4) of dimension 5 and level 23 over Q with
sign -1
> Jzero(389,2,+1);
Modular abelian variety Jzero(389) of dimension 32 and level 389
over Q with sign 1
```

0.2.2 Creating the Modular Jacobians J1(N) and JH(N)

The `Jone` command create the Jacobian of the modular curve $X_1(N)$ (which parameterizes pairs consisting of an elliptic curve and a point of order N). The command `Js` creates an abelian variety isogenous to $J_1(N)$; more precisely it is the product of abelian variteis $J_\varepsilon(N)$, where $J_\varepsilon(N)$ is the abelian variety attached to all modular forms that have character a Galois conjugate of ε. Creating $J_s(N)$ much faster than creating $J_1(N)$, since less time is spent finding the integral structure on homomology.

The `JH` command create the Jacobian $J_H(N)$ of the curve $X_H(N)$, which is the quotient of $X_1(N)$ by the subgroup H of the integers modulo N.

`JH(N, d : parameters)`

`k`	RngIntElt	*Default :* 2
`sign`	RngIntElt	*Default :* 0

Let H be some subgroup of $G = (\mathbf{Z}/N\mathbf{Z})^*$ such that G/H has order d. Create the modular abelian variety $J_H(N)$, where ϵ is a Dirichlet character mod N with kernel H and $J_H(N)$ is *isogenous* to the Jacobian of the modular curve $X_H(N)$ associated to the subgroup of $\mathrm{SL}_2(\mathbf{Z})$ of matrices $[a, b; c, d]$ with c divisible by N and a in H modulo N. It is the product of modular symbols variety $J(\epsilon)$ for all Dirichlet characters ϵ that are trivial on H.

`JH(N, gens : parameters)`

`k`	RngIntElt	*Default :* 2
`sign`	RngIntElt	*Default :* 0

Let H be the subgroup of $(\mathbf{Z}/N\mathbf{Z})^*$ generated by gens. Create the modular abelian variety $J_H(N)$, where ϵ is a Dirichlet character mod N with kernel H and $J_H(N)$ is isogenous to the Jacobian of the modular curve $X_H(N)$ associated to the subgroup of $\mathrm{SL}_2(\mathbf{Z})$ of matrices $[a, b; c, d]$ with c divisible by N and a in H modulo N. It is the product of modular symbols variety $J(\epsilon)$ for all Dirichlet characters ϵ that are trivial on H.

`Jone(N : parameters)`

`sign`	RngIntElt	*Default :* 0

`Jone(N, k : parameters)`

`sign` RngIntElt *Default :* 0

Create the modular abelian variety $J_1(N)$, i.e., the Jacobian of the modular curve $X1(N)$. Note that creating and finding the integral structure on $J_s(N)$, which is isogenous to $J_1(N)$, is much faster. Take great care in computing with $J_1(N)$, since it could be very slow.

`Jone(N, k, sign)`

`Js(N : parameters)`

`sign` RngIntElt *Default :* 0

`Js(N, k : parameters)`

`sign` RngIntElt *Default :* 0

A modular abelian variety that is **Q**-isogenous to the weight k version of $J_1(N)$. More precisely, Js is the direct sum of the modular abelian varieties attached to modular symbols spaces with Nebentypus.

Example H0E4

```
> Jone(13);
Modular abelian variety Jone(13) of dimension 2 and level 13 over Q
> Jone(13,4);
Modular motive Jone(13,4) of dimension 15 and level 13 over Q
> Jone(13,4 :  sign := 1);
Modular motive Jone(13,4) of dimension 15 and level 13 over Q
> JH(13,6);
Modular abelian variety J_H(13) of dimension 2 and level 13 over Q
> JH(13,3);
Modular abelian variety J_H(13) of dimension 0 and level 13 over Q
> JH(13,[-1]);
Modular abelian variety J_H(13) of dimension 2 and level 13 over Q
> Jone(17);
Modular abelian variety Jone(17) of dimension 5 and level 17 over Q
> Js(17);
Modular abelian variety Js(17) of dimension 5 and level 17 over Q
> IsIsogenous(Jone(17),Js(17));
true
> Degree(NaturalMap(Jone(17),Js(17)));
16
> JH(17,2);
Modular abelian variety J_H(17) of dimension 1 and level 17 over Q
> JH(17,4);
Modular abelian variety J_H(17) of dimension 1 and level 17 over Q
> JH(17,8);
```

```
Modular abelian variety J_H(17) of dimension 5 and level 17 over Q
```

0.2.3 Abelian Varieties Attached to Modular Forms

The following commands create abelian varieties attached to spaces of modular forms, sequences of spaces of forms, newforms, and characters. If an input space of modular forms is not cuspidal, MAGMA automatically replaces it with its cuspidal subspace.

`ModularAbelianVariety(M : parameters)`

`sign` RNGINTELT *Default :* 0

The abelian variety attached to the modular forms space M.

`ModularAbelianVariety(X : parameters)`

`sign` RNGINTELT *Default :* 0

The abelian variety attached to the sequence X of modular forms spaces. This is the direct sum of the spaces attached to each element of the sequence.

`ModularAbelianVariety(eps : parameters)`

`sign` RNGINTELT *Default :* 0

`ModularAbelianVariety(eps, k : parameters)`

`sign` RNGINTELT *Default :* 0

The abelian variety associated to ϵ. This corresponds to the space of modular forms of weight k and character any Galois conjugate of ϵ. We include all Galois conjugates in order to obtain an abelian variety that is defined over $\mathbf{Q}$.

`ModularAbelianVariety(f)`

The abelian variety attached to the newform f.

`Newform(A)`

A newform f so that A is isogenous to the newform abelian variety A_f. It is an error if A is not attached to a newform.

Example H0E5

We first create the modular abelian variety attached to the spaces $S_2(\Gamma_0(11))$ and $S_2(\Gamma_1(13))$. This is the direct sum of $J_0(11)$ with $J_1(13)$.

```
> X := [ModularForms(11,2), ModularForms(Gamma1(13),2)];
> A := ModularAbelianVariety(X); A;
Modular abelian variety of dimension 3 and level 11*13 over Q
> IsIsomorphic(A, Jzero(11)*Jone(13));
true Homomorphism N(1) from modular abelian variety of dimension
```

```
3 to Jzero(11) x Jone(13) (not printing 6x6 matrix)
```

Next we create the modular abelian variety A attached to $S_2(\Gamma_1(17))$ along with $J_1(17)$ and $J_s(17)$. We then note that A is isomorphic to $J_1(17)$, but there is no reason that A should be isomorphic to $J_s(17)$ (they are probably only isogenous). This example illustrates the fact that the abelian variety computed by MAGMA attached to $S_2(\Gamma_1(17))$ is $J_1(17)$ rather than $J_s(17)$. (Recall that $J_s(N)$ is a product of copies of abelian varieties corresponding to conjugacy classes of characters.)

```
> A := ModularAbelianVariety(ModularForms(Gamma1(17),2)); A;
Modular abelian variety of dimension 5 and level 17 over Q
> B := Jone(17); B;
Modular abelian variety Jone(17) of dimension 5 and level 17 over Q
> C := Js(17); C;
Modular abelian variety Js(17) of dimension 5 and level 17 over Q
> IsIsomorphic(A,B);
true Homomorphism from modular abelian variety of dimension 5 to
Jone(17) (not printing 10x10 matrix)
> Degree(NaturalMap(A,C));
16
```

If ε is a Dirichlet character and $k \geq 2$ is an integer, let S be the space of modular forms with weight k and character a Galois conjugate of ε. The command `ModularAbelianVariety(eps,k)` computes the modular abelian variety attached to S.

```
> G<eps> := DirichletGroup(13,CyclotomicField(12));
> Order(eps^2);
6
> ModularAbelianVariety(eps^2);
Modular abelian variety of dimension 2 and level 13 over Q
> ModularAbelianVariety(eps,3);
Modular motive of dimension 4 and level 13 over Q
```

Next we compute the modular abelian variety attached to a newform in $S_2(\Gamma_1(25))$.

```
> S := CuspForms(Gamma1(25),2);
> N := Newforms(S);
> #N;
2
> f := N[1][1];
> PowerSeries(f,4);
q + a*q^2 + 1/1355*(941*a^7 + 4820*a^6 + 11150*a^5 + 11522*a^4 +
    3582*a^3 + 10041*a^2 + 24432*a - 5718)*q^3 + O(q^4)
> A_f := ModularAbelianVariety(f);
> A_f;
Modular abelian variety Af of dimension 8 and level 5^2 over Q
```

The abelian variety A_f also determines the newform:

```
> PowerSeries(Newform(A_f),4);
q + a*q^2 + 1/1355*(941*a^7 + 4820*a^6 + 11150*a^5 + 11522*a^4 +
```

```
    3582*a^3 + 10041*a^2 + 24432*a - 5718)*q^3 + O(q^4)
```

The `Newform` command works even if A wasn't explicitly created using a newform.

```
> A := Decomposition(Jzero(37))[1];
> Newform(A);
q - 2*q^2 - 3*q^3 + 2*q^4 - 2*q^5 + 6*q^6 - q^7 + O(q^8)
```

0.2.4 Abelian Varieties Attached to Modular Symbols

The commands below associated modular abelian varieties to spaces of modular symbols and to sequences of spaces of modular symbols. Conversely, the associate spaces of modular symbols to modular abelian varieties. If an input space of modular symbols is not cuspidal, it is replaced by its cuspidal subspace.

`ModularAbelianVariety(M)`

The abelian variety attached to the modular symbols space M.

`ModularAbelianVariety(X)`

The abelian variety attached to the sequence X of modular symbols spaces.

`ModularSymbols(A)`

A sequence of spaces of modular symbols associated to A.

Example H0E6

We create modular abelian varieties attached to several spaces of modular symbols.

```
> M := ModularSymbols(37,2);
> ModularAbelianVariety(M);
Modular abelian variety of dimension 2 and level 37 over Q
> M := ModularSymbols(Gamma1(17));
> ModularAbelianVariety(M);
Modular abelian variety of dimension 5 and level 17 over Q
```

Note that the sign of the space of modular symbols determines the sign of the corresponding abelian variety.

```
> M := ModularSymbols(Gamma1(17),2,+1);
> A := ModularAbelianVariety(M); A;
Modular abelian variety of dimension 5 and level 17 over Q with sign 1
```

We can also create an abelian variety attached to any sequence of modular symbols spaces.

```
> ModularAbelianVariety([ModularSymbols(11), ModularSymbols(Gamma1(13))]);
Modular abelian variety of dimension 3 and level 11*13 over Q
```

Conversely, there is a sequence of modular symbols spaces associated to any abelian variety defined over $\mathbf{Q}$. These need not be the same as the spaces used to define the modular abelian variety; instead they are what is used internally in computatations on that abelian variety.

```
> ModularSymbols(Jone(13));
[
    Modular symbols space of level 13, weight 2, character $.1,
    and dimension 2 over Cyclotomic Field of order 6 and degree 2
]
> ModularSymbols(Jzero(37));
[
    Modular symbols space for Gamma_0(37) of weight 2 and
    dimension 4 over Rational Field
]
> A := Jone(17);
> ModularSymbols(A);
[
    Modular symbols space for Gamma_0(17) of weight 2 and
    dimension 2 over Rational Field,
    Modular symbols space of level 17, weight 2, character $.1,
    and dimension 2 over Cyclotomic Field of order 8 and degree 4
]
```

0.2.5 Creation of Abelian Subvarieties

Suppose A is an abelian variety and V is a vector subspace of the rational homology $H_1(A, \mathbf{Q})$. Then the `DefinesAbelianSubvariety` command determines whether or not V is the rational homology of an abelian subvariety of A, and if so computes that abelian subvariety. The `DefinesAbelianSubvariety` command relies on knowning a complete decomposition of A as a product of simple abelian varieties (so it is currently restricted to abelian varieties for which such a decomposition can be computed in Magma, e.g., all modular abelian varieties over $\mathbf{Q}$).

The other commands below are used to create the zero-dimensional abelian variety.

`DefinesAbelianSubvariety(A, V)`

True if and only if the subspace V of rational homology defines an abelian subvariety of A. If true, also returns the abelian subvariety.

`ZeroModularAbelianVariety()`

The zero-dimensional abelian variety.

`ZeroModularAbelianVariety(k)`

The zero-dimensional abelian variety of weight k.

`ZeroSubvariety(A)`

Example H0E7

We define two subspaces of the rational homology of $J_0(33)$; one defines an abelian subvariety and the other does not.

```
> A := Jzero(33);
> w3 := AtkinLehnerOperator(A,3);
> W := Kernel(Matrix(w3)+1);
> DefinesAbelianSubvariety(A,W);
true Modular abelian variety of dimension 1 and level 3*11 over Q
> V := RationalHomology(A);
> DefinesAbelianSubvariety(A,W + sub<V|[V.1]>);
false
```

We create several zero-dimensional abelian varieties.

```
> ZeroModularAbelianVariety();
Modular abelian variety ZERO of dimension 0 and level 1 over Q
> ZeroModularAbelianVariety(2);
Modular abelian variety ZERO of dimension 0 and level 1 over Q
> ZeroSubvariety(Jzero(11));
Modular abelian variety ZERO of dimension 0 and level 11 over Q
```

0.2.6 Creation Using a Label

As a useful shorthand, it is sometimes possible to create modular abelian varieties by giving a short string. If the string contains a single integer N, e.g., 37, then the corresponding abelian variety is $J_0(N)$. If it is of the form `"<level>k<weight>"`, then it is the possibly motivic $J_0(N)$ of weight k. If it is of the form `"<level>k<weight><isogeny code>"`, where `<isogeny code>` is one of `"A"`, `"B"`, ..., `"Z"`, `"AA"`, `"BB"`, ..., `"ZZ"`, `"AAA"`, `"BBB"`, ..., then the corresponding abelian variety is `Jzero(N,k)(iso)`, where iso is a positive integer, and `"A"` corresponds to iso=1, `"Z"` to iso=26, `"AA"` to iso=27, `"ZZ"` to iso=52, `"AAA"` to iso=53, etc.

This labeling convention is the same as the one used for modular symbols, and extends the one used for Cremona's database of elliptic curves, except that Cremona's database contains some random scrambling for levels between 56 and 450. If the weight part of the label is omitted, the weight is assumed to be 2. To get the optimal quotient of $J_0(N)$ with Cremona label s, set the optional parameter `Cremona` equal to true.

`ModularAbelianVariety(s : parameters)`

`Cremona` BoolElt *Default : false*

Abelian variety defined by string s.

`ModularAbelianVariety(s, sign : parameters)`

`Cremona` BoolElt *Default : false*

Abelian variety defined by string s. See the documentation for more details.

Example H0E8

```
> ModularAbelianVariety("37");
Modular abelian variety 37 of dimension 2 and level 37 over Q
> ModularAbelianVariety("37A");
Modular abelian variety 37A of dimension 1 and level 37 over Q
> ModularAbelianVariety("11k4A");
Modular motive 11k4A of dimension 2 and level 11 over Q
> ModularAbelianVariety("65C");
Modular abelian variety 65C of dimension 2 and level 5*13 over Q
> ModularDegree(ModularAbelianVariety("56A"));
4
> ModularDegree(ModularAbelianVariety("56A" :  Cremona := true));
2
```

0.2.7 Invariants

The invariants commands can be used to obtain the base ring, dimension, character of defining modular form, a field of definition, the level, the sign, the weights, and a short name of a modular abelian variety.

`BaseRing(A)`

The ring that A is defined over.

`Dimension(A)`

The dimension of A.

`DirichletCharacter(A)`

If $A = A_f$ is attached to a newform, then this returns the Nebentypus character of f. Note that since f is only well-defined up to $\mathrm{Gal}(\overline{\mathbf{Q}}/\mathbf{Q})$ conjugacy, the character is also only well-defined up to $\mathrm{Gal}(\overline{\mathbf{Q}}/\mathbf{Q})$ conjugacy.

`DirichletCharacters(A)`

List of all Dirichlet characters of spaces of modular symbols associated with the modular symbols abelian variety that parameterizes A.

`FieldOfDefinition(A)`

The best known field of definition of A.

`Level(A)`

An integer N so that A is a quotient of a power of $J_1(N)$. Note that N need not be minimal. It is determined by how A is explicitly represented as a quotient of modular Jacobians.

`Sign(A)`

The sign of A, which is either 0, -1, or $+1$. If $+1$ or -1, this means we only compute the corresponding complex-conjugation eigenspace of the homology of A, so various computations will be off by a factor of 2.

`Weights(A)`

The set of weights of A. (The weight need not be unique since direct sums of modular symbols spaces of different weights are allowed.)

Example H0E9

We illustrate all the commands for $J_0(23)$.

```
> A := Jzero(23);
> BaseRing(A);
Rational Field
> Dimension(A);
2
> DirichletCharacter(A);
1
> FieldOfDefinition(A);
Rational Field
> Level(A);
23
> Sign(A);
0
> Weights(A);
{ 2 }
```

This is an example of a nontrivial Dirichlet character.

```
> eps := DirichletCharacter(Jone(23)(2)); eps;
$.1^2
> Order(eps);
11
```

We illustrate the `Weights` command in several cases.

```
> Weights(Jzero(11));
{ 2 }
> Weights(Jzero(11,4));
{ 4 }
> Weights(Jone(13,3));
{ 3 }
> Weights(DirectSum(Jzero(11),Jone(13,3)));
{ 2, 3 }
> Weights(DirectSum(Jzero(11),Jzero(13,3)));
{ 2 }
```

We display a few fields of definition.

```
> FieldOfDefinition(Jone(13));
Rational Field
> FieldOfDefinition(BaseExtend(Jzero(11),QuadraticField(7)));
Rational Field
> FieldOfDefinition(ChangeRing(Jzero(11),GF(7)));
Finite field of size 7
```

In the following example we quotient $J_0(11)$ out by a 5-torsion point. The resulting abelian variety might not be defined over $\mathbf{Q}$, and the `FieldOfDefinition` command currently plays it safe and returns $\overline{\mathbf{Q}}$.

```
> A := Jzero(11);
> G := nTorsionSubgroup(A,5);
> H := Subgroup([G.1]);
> H;
Finitely generated subgroup of abelian variety with invariants [ 5 ]
> FieldOfDefinition(A/H);
Algebraically closed field with no variables
```

0.2.8 Conductor

Let A be a modular abelian variety over $\mathbf{Q}$. The conductor command computes the conductor of A by factoring A into newform abelian varieties and using that the conductor of A_f is N^d, where N is the level of f and d is the dimension of A_f.

`Conductor(A)`

The conductor of the abelian variety A. We require that A is defined over $\mathbf{Q}$. When $A = A_f$ is attached to a newform of level N, then the conductor of A is N^d, where d is the dimension of A.

Example H0E10

```
> Factorization(Conductor(Jzero(33)));
[ <3, 1>, <11, 3> ]
> Factorization(Conductor(Jzero(11)^5));
[ <11, 5> ]
> Factorization(Conductor(OldSubvariety(Jzero(46))));
[ <23, 4> ]
> Factorization(Conductor(Jone(25)));
[ <5, 24> ]
```

0.2.9 Number of Points

Given an abelian variety A over a field K the `NumberOfRationalPoints` or `#` command compute a divisor and multiple of $\#A(K)$. When finite, the multiple of the number of rational points is computed using reduction mod primes up to 100. Currently the lower bound is nontrivial only when A is a quotient of $J_0(N)$.

`NumberOfRationalPoints(A)`

Divisor and multiple of the cardinality of `A(K)`, where A is a modular abelian variety defined over a field K. If K is an abelian number field, then we assume the Birch and Swinnerton-Dyer conjecture.

`#A`

Same as NumberOfRationalPoints.

Example H0E11

```
> #Jzero(11);
5 5
> #Jzero(23);
11 11
> #Jzero(37);
Infinity Infinity
> #Jone(13);
1 19
> #Jone(23);
1 408991
> Factorization(408991);
[ <11, 1>, <37181, 1> ]
> NumberOfRationalPoints(ModularAbelianVariety("43B"));
7 7
```

0.2.10 Inner Twists and Complex Multiplication

If f is a newform then an *inner twist* of f is a Dirichlet character χ such that the twist of f by χ equals a Galois conjugate of f, at least at Fourier coefficients whose subscript is coprime to some fixed integer. A *CM twist* is a nontrivial character χ such that f twisted by χ equals f, at least at Fourier coefficients whose subscript is coprime to some fixed integer. The commands below find the CM and inner twists of the newform corresponding to a newform abelian variety.

The optional parameter Proof to each command is by default `false`. If `true`, it uses a huge number of terms of q-expansions to ensure that that inner twist is really an inner twist. If `false`, it uses far less (and is hence very quick), and in practice this should be OK.

`CMTwists(A : parameters)`

`Proof` BoolElt *Default : false*

The CM inner twists characters of the newform abelian variety $A = A_f$ that are defined over the base ring of A. For all CM twists, base extend to `AlgebraicClosure(RationalField())` first.

`InnerTwists(A : parameters)`

`Proof` BoolElt *Default : false*

The inner twists characters of the newform abelian variety $A = A_f$ that are defined over the base ring of A. For all CM twists, base extend to `AlgebraicClosure(RationalField())` first.

Example H0E12

We compute the inner twists for $J_1(13)$.

```
> A := Jone(13); A;
Modular abelian variety Jone(13) of dimension 2 and level 13 over Q
> CMTwists(A);
[]
> A2 := BaseExtend(A,AlgebraicClosure(RationalField()));
> CMTwists(A2);
[]
> InnerTwists(A2);
[
    1,
    $.1^5
]
> Parent($1[2]);
Group of Dirichlet characters of modulus 13 over Cyclotomic Field
of order 6 and degree 2
```

We compute the inner twists for the second newform factor of $J_1(23)$.

```
> A := Decomposition(Jone(23))[2]; A;
Modular abelian variety image(23A[2]) of dimension 10, level 23
and conductor 23^10 over Q
> InnerTwists(BaseExtend(A,AlgebraicClosure(RationalField())));
[
    1,
    $.1^20
]
```

The CM elliptic curve $J_0(32)$ has a nontrivial CM inner twist.

```
> A := Jzero(32);
> InnerTwists(BaseExtend(A,AlgebraicClosure(RationalField())));
[
    1,
```

```
    $.1
]
> CMTwists(BaseExtend(A,AlgebraicClosure(RationalField())));
[
    $.1
]
```

Quotients of $J_0(N)$ can also have nontrivial inner twists, which are not CM twists.

```
> J := Jzero(81);
> A := Decomposition(J)[1];
> InnerTwists(BaseExtend(A,AlgebraicClosure(RationalField())));
[
    1,
    $.1
]
> CMTwists(BaseExtend(A,AlgebraicClosure(RationalField())));
[]
> Newform(A);
q + a*q^2 + q^4 - a*q^5 + 2*q^7 + O(q^8)
```

The following is an example of a 4-dimensional abelian variety $A = A_f$ in $J_0(512)$ that has four inner twists, none of which are CM twists. One can use this fact to prove that if a_p is a prime-indexed Fourier coefficient of f, then $a_p^2 \in \mathbf{Z}$. Thus no single a_p generates the degree 4 field generated by all a_n.

```
> J := Jzero(512, 2, +1);
> A := Decomposition(J)[7]; A;
Modular abelian variety 512G of dimension 4, level 2^9 and
conductor 2^36 over Q with sign 1
> f := Newform(A); f;
 q + 1/12*(a^3 - 30*a)*q^3 + 1/12*(-a^3 + 42*a)*q^5 +
     1/6*(-a^2 + 18)*q^7 + O(q^8)
> Coefficient(f,3)^2;
6
> Coefficient(f,5)^2;
12
> Coefficient(f,7)^2;
8
> Abar := BaseExtend(Jzero(512,2,+1)(7),AlgebraicClosure(RationalField()));
> InnerTwists(Abar);
[
    1,
    $.1,
    $.2,
    $.1*$.2
]
> CMTwists(Abar);
[]
```

0.2.11 Predicates

Most of the predicates below work in full generality. The ones whose domain of applicability is somewhat limited are `IsIsomorphic`, `IsQuaternionic`, and `IsSelfDual`. The `IsIsomorphic` command will at least definitely determine whether any two simple modular abelian varieties over $\mathbf{Q}$ are isomorphic. In theory, it is possible to determine isomorphism for more general classes of modular abelian varieties, but this has not been implemented.

`CanDetermineIsomorphism(A, B)`

True if we can determine whether or not A and B are isomorphic. If we can determine isomorphism, also returns true if A and B are isomorphic and an explicit isomorphism, or false if they are not isomorphic. If we can not determine isomorphism, also returns the reason why we can not as a string. If A and B are simple and defined over $\mathbf{Q}$, then it is always possible to determine whether A and B are isomorphic. If one of A or B has simple factors of multiplicity one, then in principal it is possible, but the algorithm has not been programmed.

`HasMultiplicityOne(A)`

True if the simple factors of A appear with multiplicity one.

`IsAbelianVariety(A)`

True if A is an abelian variety, i.e., defined over a ring of characteristic 0 in which the conductor is invertible, or a finite field that does not divide the conductor of A. For example, if A has positive dimension and is defined over $\mathbf{Z}$, then Raynaud's theorem implies that A is not an abelian variety.

`IsAttachedToModularSymbols(A)`

True if the underlying homology of A is being computed using a space of modular symbols. For example, this will be true for $J_0(N)$ and for newform abelian varieties.

`IsAttachedToNewform(A)`

True if A is isogenous to a newform abelian variety A_f. This intrinsic also returns the abelian variety A_f. Third return argument is explicit isogeny from A_f to A.

`IsIsogenous(A, B)`

True if A and B are isogenous. If this can *not* be determined, then an error message is displayed and the program terminates. It is always possible to determine whether or not A and B are isogenous when both are defined over $\mathbf{Q}$.

`IsIsomorphic(A, B)`

True if A and B are isomorphic. If true, also returns an explicit isomorphism. This command will work if A and B are defined over $\mathbf{Q}$ and the simple factors occur with multiplicity one, and may work otherwise, but it may terminate with an error in the general case. Use the command CanDetermineIsomorphism to avoid getting an error.

`IsOnlyMotivic(A)`

True if any of the modular forms attached to A have weight bigger than 2.

`IsQuaternionic(A)`

True if and only if some simple factor of A over the base ring has quaternionic multiplication.

`IsSelfDual(A)`

True if A is known to be isomorphic to its dual. There is an error message if *Magma* is unable to decide.

`IsSimple(A)`

True if and only if A has no proper abelian subvarieties over `BaseRing(A)`.

Example H0E13

We test whether a few objects are actually abelian varieties.

```
> A := Jzero(11);
> A11 := ChangeRing(A,GF(11));
> IsAbelianVariety(A11);
false
> AZ := ChangeRing(A,Integers());
> IsAbelianVariety(AZ);
false
> A3 := ChangeRing(A,pAdicRing(3));
> IsAbelianVariety(A3);
true
```

A modular motive is sometimes also an abelian variety, but only over the complex numbers.

```
> A := Jzero(11,4); A;
Modular motive Jzero(11,4) of dimension 2 and level 11 over Q
> IsAbelianVariety(A);
false
> IsOnlyMotivic(A);
true
> IsAbelianVariety(BaseExtend(A,ComplexField()));
true
```

Abelian varieties $J_0(N)$ and $J_s(N)$ are attached to modular symbols, as are newform abelian varieties.

```
> J := Jzero(37);
```

```
> IsAttachedToModularSymbols(J);
true
> A := Decomposition(J)[1];
> IsAttachedToModularSymbols(A);
false
> t, Af := IsAttachedToNewform(A);
> IsAttachedToModularSymbols(Af);
true
> IsIsomorphic(A,Af);
true Homomorphism from 37A to 37A given on integral homology by:
[1 0]
[0 1]
> IsAttachedToModularSymbols(Js(17));
true
> IsAttachedToModularSymbols(Jone(17));
false
```

We test isogeny between a few abelian varieties.

```
> IsIsogenous(Jzero(11),Jone(11));
true
> IsIsogenous(Jzero(11)*Jzero(11),Jzero(22));
true
> IsIsogenous(Jzero(11)*Jzero(11),Jzero(33));
false
> IsIsogenous(Jzero(11),Jzero(14));
false
> IsIsogenous(Jzero(11)^2,Jzero(22));
true
> A := Jzero(37)(2); B := Jone(13);
> IsIsogenous(A*B*A, A*A*B);
true
> A := Jzero(43);
> G := RationalCuspidalSubgroup(A);
> IsIsogenous(A,A/G);
true
```

Next we test isomorphism between some abelian varieties.

```
> A := Jzero(43);
> G := RationalCuspidalSubgroup(A);
> B := A/G;
> CanDetermineIsomorphism(A,B);
true false
> IsIsomorphic(A,B);
false
> IsIsomorphic(Jzero(11),Jone(11));
false
> IsIsomorphic(Jzero(13),Jone(13));
false
```

```
> IsIsomorphic(Js(13),Jone(13));
true Homomorphism from Js(13) to Jone(13) given on integral
homology by:
[ 0  0 -1  0]
[ 0  0  0 -1]
[ 1  0 -1  0]
[ 0  1  0 -1]
> CanDetermineIsomorphism(Js(17),Jone(17));
false All tests failed to decide whether A and B are isomorphic.
```

We test whether certain Jacobians have simple factors with multiplicity one.

```
> HasMultiplicityOne(Jzero(43));
true
> HasMultiplicityOne(Jzero(33));
false
> Decomposition(Jzero(33));
[
    Modular abelian variety 33A of dimension 1, level 3*11 and
    conductor 3*11 over Q,
    Modular abelian variety N(11,33,1)(11A) of dimension 1, level
    3*11 and conductor 11 over Q,
    Modular abelian variety N(11,33,3)(11A) of dimension 1, level
    3*11 and conductor 11 over Q
]
>
```

In the next example we give an example of a non-quaternionic surface.

```
> IsQuaternionic(Jone(13));
false
```

Jacobians are self dual, and there is a surface of level 43 that is self dual and a surface of level 69 that is not.

```
> IsSelfDual(Jone(13));
true
> IsSelfDual(Jzero(69)(2));
false
```

The surface A below is isomorphic to its dual. The natural polarization has kernel that is the kernel of multiplication by 2.

```
> A  := Jzero(43)(2);
> A;
Modular abelian variety 43B of dimension 2, level 43 and
conductor 43^2 over Q
> IsSelfDual(A);
true
> phi := ModularPolarization(A);
> Invariants(Kernel(phi));
```

```
[ 2, 2, 2, 2 ]
```

We test a few abelian varieties for simplicity.

```
> IsSimple(Jone(25));
false
> IsSimple(Jone(13));
true
> IsSimple(Jzero(11)^10);
false
> IsSimple(NewSubvariety(Jzero(100)));
true
```

0.2.12 Equality and Inclusion Testing

These functions test whether two abelian varieties are exactly equal or if one is a subset of another.

`A eq B`

True if A and B are equal.

`A subset B`

True if A is a subset of B.

Example H0E14

This example illustrates that taking the direct product of abelian varieties is not commutative, in the sense of equality (though it is in the sense of isomorphism).

```
> A := Jzero(11);
> B := Jzero(14);
> A*B eq A*B;
true
> A*B eq B*A;
false
> IsIsomorphic(A*B, B*A);
true Homomorphism N(1) from Jzero(11) x Jzero(14) to Jzero(14) x Jzero(11)
given on integral homology by:
[0 0 1 0]
[0 0 0 1]
[1 0 0 0]
[0 1 0 0]
```

The first inclusion below is as expected, but the second non-inclusion might be surprising. We do not consider $J_0(11)$ as a subset of $J_0(22)$, even though there is an injective map from one to the other, since to be a subset is much stronger than just the existence of an inclusion map.

```
> Jzero(37) subset Jzero(37);
```

```
true
> Jzero(11) subset Jzero(22);
false
> IsInjective(NaturalMap(Jzero(11),Jzero(22)));
true
```

0.2.13 Modular Embedding and Parameterization

Every modular abelian variety A is equipped with a modular parameterization and a modular embedding. The modular parameterization is a surjective homomorphism from a modular symbols abelian variety, such as $J_0(N)$. The modular embedding is a homomorphism to a modular symbols abelian variety, which is only guaranteed to be *injective in the category of abelian varieties up to isogeny*. The structure of these two homomorphisms is extremely important as it is completely defines A.

`CommonModularStructure(X)`

This intrinsic finds modular abelian varieties J_e and J_p associated to modular symbols and returns a list of finite-kernel maps from the abelian varieties in X to J_e and a list of modular paramaterizations from J_p to the abelian varieties in X.

`ModularEmbedding(A)`

A morphism with finite kernel from A to a modular abelian variety attached to modular symbols. This is only guaranteed to be an embedding in the category of abelian varieties up to isogeny.

`ModularParameterization(A)`

A surjective morphism to A from an abelian variety attached to modular symbols.

Example H0E15

```
> X := [Jzero(11),ModularAbelianVariety("37B")];
> CommonModularStructure(X);
[*
Homomorphism from Jzero(11) to Jzero(11) x Jzero(37) given on integral
homology by:
[1 0 0 0 0 0]
[0 1 0 0 0 0],
Homomorphism from 37B to Jzero(11) x Jzero(37) given on integral
homology by:
[0 0 1 1 1 0]
[0 0 0 0 0 1]
*] [*
Homomorphism from Jzero(11) x Jzero(37) to Jzero(11) (not printing 6x2
matrix),
```

```
Homomorphism from Jzero(11) x Jzero(37) to 37B (not printing 6x2
matrix)
*]
```

The next example illustrates that the modular "embedding" need only be an embedding in the category of abelian varieties up to isogeny.

```
> A := Jzero(37)(1);
> x := A![1/2,1];
> B := A/Subgroup([x]);
> e := ModularEmbedding(B);
> e;
Homomorphism from modular abelian variety of dimension 1 to
Jzero(37)_Qbar given on integral homology by:
[ 1 -1  1  0]
[ 2 -2 -2  2]
> IsInjective(e);
false
```

Moreover, the modular parameterization is surjective, but it need be optimal (have connected kernel).

```
> pi := ModularParameterization(B);
> IsSurjective(pi);
true
> ComponentGroupOfKernel(pi);
Finitely generated subgroup of abelian variety with invariants [ 2 ]
> IsOptimal(pi);
false
```

0.2.14 Coercion

Coercion can be used to create points on modular abelian varieties from vectors on a basis of integral homology, from other elements of modular abelian varieties, or from modular symbols. See the examples below, especially the last one, to understand some of the subtleties of coercision that arise because we view an abelian variety as a vector space modulo a lattice, and the lattice can be embedded in any way in $\mathbf{Q}^n$.

`A ! x`

Coerce x into A. The argument x can be an element of a modular abelian variety, the integer 0, a sequence obtained by Eltseq'ing an element of A (i.e., a linear combination of *integral* homology), a vector on the basis for *rational* homology, or a tuple of the form $< P(X,Y), [u,v] >$ that defines a modular symbol.

Example H0E16

If you coerce a *sequence* of rationals or reals into an abelian variety A, then MAGMA computes the corresponding linear combination of a basis of integral homology and returns the point it defines. The sequence must have length the rank of the integral homology.

```
> Jzero(11)![1/2,1/5];
Element of abelian variety defined by [1/2 1/5] modulo homology
```

If you coerce exactly two cusps (or extended reals) into A, then MAGMA computes the point corresponding to that modular symbol.

```
> Jzero(11)![Cusps()|1/2,1/5];
0
> Jzero(11)![Sqrt(2),0];
Element of abelian variety defined by
[1.41421356237309504880168872419 8] modulo homology
> Jzero(11)![Cusps()|0,Infinity()];     // cusps
Element of abelian variety defined by [0 1/5] modulo homology
> Jzero(11)![0,Infinity()];             // extended reals
Element of abelian variety defined by [0 1/5] modulo homology
```

Coercion of modular symbols also works for higher weight.

```
> Jzero(11,4)![0,Infinity()];
Element of abelian variety defined by [-4/61 5/61 1/61 -1/61]
modulo homology
> R<x,y> := PolynomialRing(RationalField(),2);
> Jzero(11,4)!<x^2,[0,Infinity()]>;
Element of abelian variety defined by [-4/61 5/61 1/61 -1/61]
modulo homology
> Jzero(11,4)!<y^2,[0,Infinity()]>;
Element of abelian variety defined by [44/61 -55/61 -11/61 11/61]
modulo homology
```

You can also coerce elements from abelian subvarieties into an ambient abelian variety.

```
> J := Jzero(37); A := Decomposition(J)[1];
> x := A![1/5,0];
> Parent(x);
Modular abelian variety 37A of dimension 1, level 37 and
conductor 37 over Q
> x in J;
false
> y := J!x; y;
Element of abelian variety defined by [1/5 -1/5 1/5 0] modulo
homology
> y in J;
true
> Parent(y);
Modular abelian variety Jzero(37) of dimension 2 and level 37 over Q
```

Coercion also provides an easy way to create the 0 element.

```
> Jzero(37)!0;
0
```

The following example illustrates the subtlety of coercion when the element being coerced in is a vector instead of a sequence. We create the quotient of $J_0(11)$ by a cyclic subgroup of order 10. The lattice that defines $J_0(11)$ is $\mathbf{Z} \times \mathbf{Z}$, but the lattice that defines this quotient is $(1/10)\mathbf{Z} \times \mathbf{Z}$. Thus the natural quotient map on vector spaces is defined by the identity matrix. On the other hand, the matrix of the quotient with respect to a basis for integral homology has determinant 10.

```
> A := Jzero(11);
> x := A![1/10,0]; x;
Element of abelian variety defined by [1/10 0] modulo homology
> Order(x);
10
> B,pi := A/Subgroup([x]);
> B;
Modular abelian variety of dimension 1 and level 11 over Qbar
> pi;
Homomorphism from Jzero(11)_Qbar to modular abelian variety of
dimension 1 given on integral homology by:
[10  0]
[ 0  1]
> Matrix(pi);
[1 0]
[0 1]
> IntegralMatrix(pi);
[10  0]
[ 0  1]
> base := Basis(IntegralHomology(B)); base;
[
    (1/10    0),
    (0 1)
]
```

If we coerce in the sequence [1/10,0] we get the point in B that is represented by 1/10th of the first generator for homology. If we coerce in the vector $(1/10, 0)$, we instead get the element of B represented by that element of the rational homology, which is 0, since the lattice that defines B is embedded in such a way that it contains $(1/10, 0)$.

```
> y := B![1/10,0]; y;
Element of abelian variety defined by [1/10 0] modulo homology
> Order(y);
10
> z := B!base[1]; z;
0
```

0.2.15 Modular Symbols to Homology

Modular symbols determine elements of the rational homology of $J_0(N)$, $J_1(N)$, etc., and hence of arbitrary modular abelian varieties, by using the modular parameterization. The commands below convert from modular symbols, which are represented in various ways, to vectors on the basis for rational or integral homology.

`ModularSymbolToIntegralHomology(A, x)`

`ModularSymbolToIntegralHomology(A, x)`

The element of integral homology naturally associated to the (formal) modular symbol $s = P(X,Y)\{\alpha,\beta\}$, where α, β are in $P^1(\mathbf{Q})$ and P is a homogeneous polynomial of degree 2. The returned vector is written with respect to the basis of integral homology. This intrinsic takes its input a sequence $[a,b]$ or a pair $< P(X,Y),[a,b] >$, where a, b are in `Cusps()`.

`ModularSymbolToRationalHomology(A, x)`

`ModularSymbolToRationalHomology(A, x)`

`ModularSymbolToRationalHomology(A, x)`

The element of rational homology naturally associated to the (formal) modular symbol $s = P(X,Y)\{\alpha,\beta\}$, where α, β are in $P^1(\mathbf{Q})$ and P is a homogeneous polynomial of degree 2. The returned vector is written with respect to the basis of rational homology. This intrinsic takes its input as a pair $< P(X,Y),[c,d] >$, where c, d are in `Cusps()`.

Example H0E17

```
> A := Jzero(11);
> x := ModularSymbolToIntegralHomology(A,[0,Infinity()]); x;
(  0 1/5)
> z := A!x; z;
Element of abelian variety defined by [0 1/5] modulo homology
> Order(z);
5
> A := Jzero(47);
> x := ModularSymbolToIntegralHomology(A,[0,Infinity()]); x;
(-1/23  3/23 -4/23  4/23 -3/23  1/23  2/23  8/23)
> z := A!x;
> Order(z);
23

> J := Jzero(11,4);
> IntegralHomology(J);
Lattice of rank 4 and degree 4
```

```
Basis:
( 3  1 -1 -1)
( 1 -1 -3  1)
( 2 -2  2  2)
( 2 -2  2 -2)
Basis Denominator:  8
> ModularSymbolToIntegralHomology(J,[0,Infinity()]);
(-4/61  5/61  1/61 -1/61)
```

Notice that for weight greater than 2 the homomogenous polynomial part of the modular symbol may be omitted. If so, it defaults to x^{k-2}.

```
> R<x,y> := PolynomialRing(RationalField(),2);
> ModularSymbolToIntegralHomology(J,<x^2,[0,Infinity()]>);
(-4/61  5/61  1/61 -1/61)
> ModularSymbolToIntegralHomology(J,<y^2,[0,Infinity()]>);
( 44/61 -55/61 -11/61  11/61)
```

The result of coercion to rational homology is different because it is written in terms of the basis for rational homology instead of the basis for integral homology, and in this example the two basis differ.

```
> ModularSymbolToRationalHomology(J,[0,Infinity()]);
( -7/488  -9/488 -11/488  13/488)
```

Coercion is also a way to define torsion points on abelian varieties.

```
> Jzero(37)![1/5,0,0,0];
Element of abelian variety defined by [1/5 0 0 0] modulo homology
```

0.2.16 Embeddings

The `Embeddings` command contains a list of embeddings (up to isogeny) from A to other abelian varieties. The `AssertEmbedding` command allows you to add an embedding to the beginning of the list. The embeddings are used for computing intersections, sums, etc., with the embedding at the front of the list having highest priority.

`AssertEmbedding( A, phi)`

Place ϕ at the beginning of `Embeddings(A)`. The morphism ϕ must have finite kernel.

`Embeddings(A)`

A list of morphisms from A into abelian varieties, which are used in making sense of intersections, sums, etc. The embeddings at the beginning of the list take precedence over those that occur later. Note that these maps might not really be injective; e.g., the modular embedding, which need only be injective on homology, is always at the end of this list.

Example H0E18

Every modular abelian variety comes equipped with at least one embedding, the "modular embedding".

```
> Embeddings(Jzero(11));
[*
Homomorphism from Jzero(11) to Jzero(11) given on integral homology by:
[1 0]
[0 1]
*]
> A := Jzero(37)(1);
> Embeddings(A);
[*
Homomorphism from 37A to Jzero(37) given on integral homology by:
[ 1 -1  1  0]
[ 1 -1 -1  1]
*]
```

We add another embedding to the list of embeddings for A.

```
> phi := NaturalMap(A,Jzero(37*2));
> AssertEmbedding(~A,phi);
> Embeddings(A);
[*
Homomorphism N(1) from 37A to Jzero(74) given on integral homology
by:
[-1  1 -1  0  2 -1  1  0 -2  1 -2  2 -1  2 -1  1]
[-1  0  0  0  2 -1  1 -2  0  0 -1  1 -1  2  1 -1],
Homomorphism from 37A to Jzero(37) given on integral homology by:
[ 1 -1  1  0]
[ 1 -1 -1  1]
*]
```

The following intersection would not make sense if we hadn't chosen an embedding of A into $J_0(74)$.

```
> B := Codomain(phi)(1); B;
Modular abelian variety 74A of dimension 2, level 2*37 and
conductor 2^2*37^2 over Q
> #(A meet B);
1
```

0.2.17 Base Change

The `BaseExtend` and `ChangeRing` commands allow you to change the base ring of a modular abelian variety. The `BaseExtend` command is the same `ChangeRing`, but is more restrictive. For example, if A is over $\mathbf{Q}$ then `BaseExtend(A,GF(2))` is not allowed, but `ChangeRing(A,GF(2))` is.

Abelian varieties can have their base ring set to a finite field, but there is very little that is implemented for abelian varieties over finite fields. Computing the number of points is implemented, but creation of actual points or homomorphisms is not.

`BaseExtend(A, R)`

Extend the base ring of A to R, if possible. This is merely a more restrictive version of ChangeRing.

`CanChangeRing(A, R)`

True if it is possible to change the base ring of A to R, and A over R when possible.

`ChangeRing(A, R)`

Change the base ring of A to R, if possible.

Example H0E19

We consider $J_(13)$ over many fields and rings.

```
> A := Jone(13);
> BaseExtend(A,CyclotomicField(7));
Modular abelian variety Jone(13) of dimension 2 and level 13 over
Q(zeta_7)
> BaseExtend(A,AlgebraicClosure(RationalField()));
Modular abelian variety Jone(13)_Qbar of dimension 2 and level 13
over Qbar
> BaseExtend(A,RealField());
Modular abelian variety Jone(13)_R of dimension 2 and level 13 over R
> BaseExtend(A,ComplexField());
Modular abelian variety Jone(13)_C of dimension 2 and level 13 over C
> ChangeRing(A,GF(3));
Modular abelian variety Jone(13)_GF(3) of dimension 2 and level 13
over GF(3)
> #ChangeRing(A,GF(3));
19 19
> B := ChangeRing(A,GF(13)); B;
Modular abelian variety Jone(13)_GF(13) of dimension 2 and level 13
over GF(13)
> IsAbelianVariety(B);
false
> ChangeRing(A,Integers());
Modular abelian variety Jone(13)_Z of dimension 2 and level 13 over
Z
> ChangeRing(A,PolynomialRing(RationalField(),10));
```

```
Modular abelian variety Jone(13) of dimension 2 and level 13 over
Polynomial ring of rank 10 over Rational Field
Lexicographical Order
Variables:  $.1, $.2, $.3, $.4, $.5, $.6, $.7, $.8, $.9, $.10
```

0.2.18 Additional Examples

Example H0E20

The simplest abelian variety is an abelian variety of dimension 0, i.e., a point.

```
> A := ZeroModularAbelianVariety(); A;
Modular abelian variety ZERO of dimension 0 and level 1 over Q
```

We create the Jacobian $J_0(22)$ of the modular curve $X_0(22)$ as follows:

```
> J := Jzero(22); J;
Modular abelian variety Jzero(22) of dimension 2 and level 2*11 over Q
```

Notice that A is a subset of $J_0(22)$.

```
> A subset J;
true
```

We can also create the higher weight analogues $J_0(N, k)$ of $J_0(N)$, which are motives defined over $\mathbf{Q}$. Many computations that make sense for $J_0(N)$ also make sense for these higher weight analogues.

```
> J4 := Jzero(22,4); J4;
Modular motive Jzero(22,4) of dimension 7 and level 2*11 over Q
> IsOnlyMotivic(J4);
true
```

One can also create $J_1(N)$:

```
> Jone(22);
Modular abelian variety Jone(22) of dimension 6 and level 2*11 over Q
```

For efficiency purposes, it is often much quicker to do computations working only with the $+1$ quotient of $H_1(J_0(N), \mathbf{Z})$ by $* = 1$, or using only the -1 quotient. These computations will be off by powers of 2, or subgroups will be halved and off by a power of 2. Nonetheless, if you know what you are doing, such computations can be very useful. Create $J_0(N)$, but working only with the $+1$ quotient of homology as follows:

```
> Jplus := Jzero(22,2,+1); Jplus;
Modular abelian variety Jzero(22) of dimension 2 and level 2*11 over Q
with sign 1
> Sign(Jplus);
1
> Sign(Jzero(22));
0
```

Notice that the sign is printed out when it is 1 or -1. Also, sign 0 is shorthand for "no sign", i.e., working with the full homology. It is not possible to take the direct sum of abelian varieties with different signs.
Let $\varepsilon : (\mathbf{Z}/N\mathbf{Z})^* \to \mathbf{Q}(\zeta_n)^*$ be Dirichlet character. The command `ModularAbelianVariety(eps)` creates the modular abelian variety *over* $\mathbf{Q}$ corresponding to the cusp forms with Dirichlet character any Galois conjugate of ε.

```
> G<eps> := DirichletGroup(22,CyclotomicField(EulerPhi(22)));
> Order(eps);
10
> Conductor(eps);
11
> A := ModularAbelianVariety(eps); A;
Modular abelian variety of dimension 0 and level 2*11 over Q
> A := ModularAbelianVariety(eps^2); A;
Modular abelian variety of dimension 4 and level 2*11 over Q
```

Let H be a sugroup of $G = (\mathbf{Z}/N\mathbf{Z})^*$. The group $(\mathbf{Z}/N\mathbf{Z})^*$ acts by diamond bracket operators as a group of automorphisms on $X_1(N)$, and the modular curve $X_H(N)$ is the quotient of $X_1(N)$ by the action of H. Thus if $H = 1$, then $X_H(N) = X_1(N)$, and if $H = G$, then $X_H(N) = X_0(N)$. The command `JH(N,d)` creates an abelian variety *isogenous to* the Jacobian of $X_H(N)$, where d is the index of H in G (thus $d = 1$ corresponds to $J_0(N)$). A weight k and sign (either 0 or ± 1) can be provided as optional parameters. More precisely, `JH(N,d)` is the product of $J(\varepsilon)$, where ε varies over Dirichlet characters such that $\varepsilon(H) = \{1\}$.

```
> JH(22,1);
Modular abelian variety Jzero(22) of dimension 2 and level 2*11 over Q
> JH(22,10);
Modular abelian variety Js(22) of dimension 6 and level 2*11 over Q
> JH(22,2);
Modular abelian variety J_H(22) of dimension 2 and level 2*11 over Q
```

As a shortcut, the command `Js(N)` creates a modular abelian variety that is isogenous to $J_1(N)$. Thus `Js(N)` is the same as JH(N,d), where d is the order of $(\mathbf{Z}/N\mathbf{Z})^*$.

```
> Js(22);
Modular abelian variety Js(22) of dimension 6 and level 2*11 over Q
> JH(22,10) eq Js(22);
true
```

We can also create modular abelian varieties attached to spaces of modular forms and spaces of modular symbols:

```
> ModularAbelianVariety(ModularForms(22));
Modular abelian variety of dimension 2 and level 2*11 over Q
> ModularAbelianVariety(ModularSymbols(22));
Modular abelian variety of dimension 2 and level 2*11 over Q
```

Here is another example, in which the space of modular forms is for $\Gamma_1(N)$.

```
> M := ModularForms(Gamma1(25)); M;
Space of modular forms on Gamma_1(25) of weight 2 and dimension
```

```
39 over Integer Ring.
> S := CuspidalSubspace(M); S;
Space of modular forms on Gamma_1(25) of weight 2 and dimension
12 over Integer Ring.
> A := ModularAbelianVariety(S); A;
Modular abelian variety of dimension 12 and level 5^2 over Q
```

We can also construct abelian varieties attached to newforms.

```
> S := CuspForms(43);
> N := Newforms(S); N;
[* [*
q - 2*q^2 - 2*q^3 + 2*q^4 - 4*q^5 + 4*q^6 + O(q^8)
*], [*
q + a*q^2 - a*q^3 + (-a + 2)*q^5 - 2*q^6 + (a - 2)*q^7 + O(q^8),
q + b*q^2 - b*q^3 + (-b + 2)*q^5 - 2*q^6 + (b - 2)*q^7 + O(q^8)
*] *]
> f := N[2][1]; f;
q + a*q^2 - a*q^3 + (-a + 2)*q^5 - 2*q^6 + (a - 2)*q^7 + O(q^8)
> A := ModularAbelianVariety(f); A;
Modular abelian variety Af of dimension 2 and level 43 over Q
> Newform(A);
q + a*q^2 - a*q^3 + (-a + 2)*q^5 - 2*q^6 + (a - 2)*q^7 + O(q^8)
```

When possible, we can also obtain a newform that gives rise to an abelian variety.

```
> J := Jzero(43);
> D := Decomposition(J); D;
[
    Modular abelian variety 43A of dimension 1, level 43 and
    conductor 43 over Q,
    Modular abelian variety 43B of dimension 2, level 43 and
    conductor 43^2 over Q
]
> Newform(D[2]);
q + a*q^2 - a*q^3 + (-a + 2)*q^5 - 2*q^6 + (a - 2)*q^7 + O(q^8)
```

Modular abelian varieties may be described by a label, which is a string like "43B". Continuing the previous code, we have:

```
> A := ModularAbelianVariety("43B");
> A eq D[2];
true
```

Notice that we created A above using the command `ModularAbelianVariety` with a string as an argument. This is only guaranteed to work on labels of the form a level followed by an isogeny code.

The integral, rational and real homology of a modular abelian variety A is $H_1(A, R)$, where $R = \mathbf{Z}, \mathbf{R}, \mathbf{Q}$, respectively. Thus this homology is just a free module over R of dimension twice $\dim(A)$ (unless the sign of A is ± 1, in which case the homology is of dimension $\dim(A)$).

```
> J := Jzero(22); J;
```

```
Modular abelian variety Jzero(22) of dimension 2 and level 2*11 over Q
> IntegralHomology(J);
Standard Lattice of rank 4 and degree 4
> RationalHomology(J);
Full Vector space of degree 4 over Rational Field
> RealHomology(J);
Full Vector space of degree 4 over Real Field
> J := Jzero(22, 2, +1); J;
Modular abelian variety Jzero(22) of dimension 2 and level 2*11 over Q
with sign 1
> RationalHomology(J);
Full Vector space of degree 2 over Rational Field
```

Using the product operator we can take the direct product of any two modular abelian varieties, even ones of different weights or levels. We first illustrate taking the product of two abelian subvarieties of $J_0(65)$, then taking the product of one of the subvarieties of $J_0(65)$ with the weight 4 motive $J_1(11, 4)$.

```
> J := Jzero(65);
> D := Decomposition(J); D;
[
    Modular abelian variety 65A of dimension 1, level 5*13 and
    conductor 5*13 over Q,
    Modular abelian variety 65B of dimension 2, level 5*13 and
    conductor 5^2*13^2 over Q,
    Modular abelian variety 65C of dimension 2, level 5*13 and
    conductor 5^2*13^2 over Q
]
> A := D[1];
> B := D[2];
> A*B;
Modular abelian variety 65A x 65B of dimension 3 and level 5*13
over Q
Homomorphism from 65A to 65A x 65B given on integral homology by:
[1 0 0 0 0 0]
[0 1 0 0 0 0]
Homomorphism from 65B to 65A x 65B given on integral homology by:
[0 0 1 0 0 0]
[0 0 0 1 0 0]
[0 0 0 0 1 0]
[0 0 0 0 0 1]
Homomorphism from 65A x 65B to 65A (not printing 6x2 matrix)
Homomorphism from 65A x 65B to 65B (not printing 6x4 matrix)
> M := Jzero(11,4);M;
Modular motive Jzero(11,4) of dimension 2 and level 11 over Q
> P := A*M; P;
Modular motive 65A x Jzero(11,4) of dimension 3 and level 5*11*13 over Q
```

The product also returns inclusions of each factor into the product and projection from the product onto each factor.

```
> C,f,g,h,k := A*B;
> f;
Homomorphism from 65A to 65A x 65B given on integral homology by:
[1 0 0 0 0 0]
[0 1 0 0 0 0]
```

The command `DirectSum`, which takes either two arguments, or a sequence of modular abelian varieties, can be used to compute arbitrary finite direct sums. Note that the `+` operator applied to two modular abelian varieties is not the direct sum. It is the sum of the two abelian varieties in a common ambient abelian variety. Thus if A is as above, then

```
> Dimension(A);
1
> Dimension(A*A);
2
> Dimension(A+A);
1
```

If you take a direct sum of abelian varieties that are defined over different base rings, then the program will first attempt to coerce them to a common over-ring.

```
> A := Jzero(11);
> B := BaseExtend(Jzero(14),CyclotomicField(3));
> C := A*B; C;
Modular abelian variety Jzero(11) x Jzero(14) of dimension 2 and level
2*7*11 over Q(zeta_3)
```

The above would not work if `CyclotomicField(3)` were replaced by `GF(3)`, since the base ring $\mathbf{Q}$ of A is not contained in `GF(3)`.

0.3 Homology

The homology $H_1(A, R)$ with coefficients in R of an abelian variety A is a free R-module of rank equal to twice the dimension of A. For many purposes, we view abelian varieties as complex tori V/Λ, and then there is a canonical isomorphism $\Lambda \cong H_1(A, \mathbf{Z})$. (If the sign of A is ± 1, then the homology command below gives a $\mathbf{Z}$-module of rank $\dim A$.)

0.3.1 Creation

The `Homology` command creates the first homology of a modular abelian variety, which is of type `ModAbVarHomol`. This is the only command for creating homology.

`Homology(A)`

The first integral homology of A. (If the sign of A is 1 or -1, then this is $\mathbf{Z}$-module of rank equal to the dimension of A.)

Example H0E21

The homology of the elliptic curve $J_0(14)$ if of dimension 2.

```
> A := Jzero(14); A;
Modular abelian variety Jzero(14) of dimension 1 and level 2*7 over Q
> Homology(A);
Modular abelian variety homology space of dimension 2
```

If for efficiency purposes we work in the $+1$ quotient, then we are only working with half the homology, so the dimension of the homology is 1.

```
> Homology(Jzero(14,2,+1));
Modular abelian variety homology space of dimension 1
```

0.3.2 Invariants

The only invariant of homology is its dimension. If A is an abelian variety, and H is its first homology, then H has dimension equal to twice the dimension of A. (Except if, for efficiency purposes, we are working with sign $+1$ or -1, in which case the dimension of H is equal to the dimension of A.)

`Dimension(H)`

The dimension of the space H of homology.

Example H0E22

```
> Homology(Jzero(100));
Modular abelian variety homology space of dimension 14
> Homology(Jzero(100,2,+1));
Modular abelian variety homology space of dimension 7
> Homology(Jzero(100,2,-1));
Modular abelian variety homology space of dimension 7
```

0.3.3 Functors to categories of lattices and vector spaces

The following commands provide convenient functors from the categories of homology and modular abelian varieties to lattices and vector spaces. Mathematically, these functors are defined using the first homology of the complex manifold underlying the complex points of the abelian variety.

`IntegralHomology(A)`

The lattice underlying the homology of A.

`Lattice(H)`

The underlying lattice of the homology space H. This is a free $\mathbf{Z}$-module of rank equal to the dimension of H.

`RationalHomology(A)`

A $\mathbf{Q}$-vector space obtained by tensoring the homology of A with $\mathbf{Q}$.

`RealHomology(A)`

A vector space over R obtained by tensoring the homology of A with R, where R is the field of real numbers.

`RealVectorSpace(H)`

The $\mathbf{R}$-vector space of dimension equal to the dimension of H.

`VectorSpace(H)`

The $\mathbf{Q}$-vector space of dimension equal to the dimension of H.

Example H0E23

The following code demonstrates the above commands applied to the abelian surface attached to the space of cusp forms of level 37 with quadratic character.

```
> G<eps> := DirichletGroup(37);
> Order(eps);
2
> A := ModularAbelianVariety(eps); A;
Modular abelian variety of dimension 2 and level 37 over Q
> Decomposition(A);
[
    Modular abelian variety image(37A[18]) of dimension 2, level
    37 and conductor 37^2 over Q
]
> IntegralHomology(A);
Standard Lattice of rank 4 and degree 4
> H := Homology(A); H;
Modular abelian variety homology space of dimension 4
> Lattice(H);
Standard Lattice of rank 4 and degree 4
> RationalHomology(A);
Full Vector space of degree 4 over Rational Field
> RealHomology(A);
Full Vector space of degree 4 over Real Field
> RealVectorSpace(H);
Full Vector space of degree 4 over Real Field
```

```
> VectorSpace(H);
Full Vector space of degree 4 over Rational Field
```

The code below illustrates that the lattice need not just be the free lattice on $\dim(H)$ generators. The reason is because for efficiency reasons many internal computations only require knowing $H_1(A, \mathbf{Q})$, which is sometimes all that gets computed, and the lattice returned by this command is $H_1(A, \mathbf{Z})$ written with respect to a basis for $H_1(A, \mathbf{Q})$. Thus `Lattice` should be viewed as giving an integral structure to $H_1(A, \mathbf{Q})$.

```
> J := Jzero(37);
> A := J(1);
> Lattice(Homology(A));
Lattice of rank 2 and degree 2
Basis:
( 1  1)
( 1 -1)
> IntegralHomology(Jone(17));
Lattice of rank 10 and degree 10
Basis:
( 0  0  1  0  0  0  1  0  0  0)
( 0  0  1  0  0  0  0  0 -1  0)
( 0  0  0  1  0  0  0  1  0  0)
( 0  0  0  1  0  0  0 -1  0  0)
( 0  0  0  0  1  0  0  0 -1  0)
( 0  0  0  0  0  1  0  0  0  1)
( 0  0  0  0  0  1  0  0  0 -1)
( 0  0  0  0  0  0  1  0 -1  0)
( 0  2  1  0  0  0  0  0  0  0)
( 2 -1  0  0  0  0  1 -1  0  1)
```

0.3.4 Modular structure

Is H is the homology of an abelian variety that is attached to a space of modular symbols, then H remembers that space of modular symbols. The following two functions allow you to decide if H is attached to modular symbols, and if so obtain the corresponding spaces of modular symbols. The reason that `ModularSymbols` returns a sequence instead of a single modular symbols space is that arbitrary finite products of abelian varieties are allowed, so corresponding direct sums of modular symbols spaces must be allowed, which are represented as sequences because MAGMA currently doesn't have a facility for taking direct sums of modular symbols spaces.

`IsAttachedToModularSymbols(H)`

True if H is presented as being attached to a sequence of spaces of modular symbols.

`ModularSymbols(H)`

If H is attached to a sequence of spaces of modular symbols, then this is that sequence. Otherwise calling this results in an error.

Example H0E24

In this example, we create the product $J_0(23) \times J_0(11)$, and find that its homology is attached modular symbols, since both $J_0(23)$ and $J_0(11)$ are attached to modular symbols. We then display the corresponding spaces of modular symbols.

```
> A := Jzero(23) * Jzero(11);
> H := Homology(A); H;
Modular abelian variety homology space of dimension 6
> IsAttachedToModularSymbols(H);
true
> ModularSymbols(H);
[
    Modular symbols space for Gamma_0(23) of weight 2 and
    dimension 4 over Rational Field,
    Modular symbols space for Gamma_0(11) of weight 2 and
    dimension 2 over Rational Field
]
```

0.3.5 Additional Examples

Example H0E25

The following example illustrates each of the above intrinsics for the homology of $J_0(23)$.

```
> A := Jzero(23); A;
Modular abelian variety Jzero(23) of dimension 2 and level 23 over Q
> H := Homology(A); H;
Modular abelian variety homology space of dimension 4
```

Notice that homology has its own type:

```
> Type(H);
ModAbVarHomol
```

In this case, the homology is attached to a space of modular symbols, since $J_0(23)$.

```
> IsAttachedToModularSymbols(H);
true
> ModularSymbols(H);
[
    Modular symbols space for Gamma_0(23) of weight 2 and
    dimension 4 over Rational Field
]
> Dimension(H);
4
> Lattice(H);
Standard Lattice of rank 4 and degree 4
> RealVectorSpace(H);
```

```
Full Vector space of degree 4 over Real Field
> VectorSpace(H);
Full Vector space of degree 4 over Rational Field
```

We can also obtain various homologies of A more directly.

```
> IntegralHomology(A);
Standard Lattice of rank 4 and degree 4
> RationalHomology(A);
Full Vector space of degree 4 over Rational Field
> RealHomology(A);
Full Vector space of degree 4 over Real Field
```

The following example illustrates that a factor of $J_0(37)$ is not presented as something associated to a space of modular symbols. Also notice that the two lattices are different.

```
> J := Jzero(37);
> D := Decomposition(J);
> H1 := Homology(D[1]);  H2 := Homology(D[2]);
> IsAttachedToModularSymbols(H1);
false
> Lattice(H1);
Lattice of rank 2 and degree 2
Basis:
( 1  1)
( 1 -1)
> Lattice(H2);
Standard Lattice of rank 2 and degree 2
```

0.4 Homomorphisms

0.4.1 Creation

The commands below create the multiplication by n map on abelian variety, for any n. Other ways to create homomorphisms are described in other sections of this chapter. These include computing Hecke operators, Atkin-Lehner operators, and computing the full endomorphism or homomorphism rings, and choosing elements in it.

`IdentityMap(A)`

The identity homomorphism from A to A.

`ZeroMap(A)`

The zero homomorphism from A to A.

`nIsogeny(A, n)`

The multiplication by n isogeny on A.

`nIsogeny(A, n)`

The multiplication by n isogeny on A.

Example H0E26

```
> A := Jzero(23);
> IdentityMap(A);
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
> ZeroMap(A);
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[0 0 0 0]
[0 0 0 0]
[0 0 0 0]
[0 0 0 0]
> nIsogeny(A,3);
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[3 0 0 0]
[0 3 0 0]
[0 0 3 0]
[0 0 0 3]
> nIsogeny(A,1/3);
Homomorphism from Jzero(23) to Jzero(23) (up to isogeny) on integral
homology by:
[1/3   0   0   0]
[  0 1/3   0   0]
[  0   0 1/3   0]
[  0   0   0 1/3]
```

0.4.2 Restriction, evaluation, and other manipulations

Suppose A is an abelian variety, B is an abelian subvariety, and ϕ is a homomorphism from A. Then the `Restriction` commands computes the restriction of ϕ to B. If moreover ϕ is an endomorphism of A (i.e., has codomain A), and $\phi(B)$ is contained in B, then `RestrictEndomorphism` computes the endomorphism of B induced by ϕ.

If f is a polynomial over the integers or rational numbers and ϕ is an endomorphism of an abelian variety, then the `Evaluate` command computes the endomorphism $f(\phi)$ (if f has denominators, then $f(\phi)$ might only be a morphism on abelian varieties up to isogeny). Together with the `Kernel` command, `Evaluate` is useful for cutting out subvarieties of an abelian variety.

The `SurjectivePart`, `DivideOutIntegers`, and `UniversalPropertyOfCokernel` commands can also all be useful in various contexts for constructing homomorphisms.

`DivideOutIntegers(phi)`

If $\phi : A \to B$ is a homomorphism of abelian varieties, find the largest integer n such that $\psi = (1/n) * \phi$ is also a homomorphism from A to B and return ψ and n.

`Evaluate(f, phi)`

The endomorphism $f(\phi)$ of A, where f is a univariant polynomial and ϕ is an endomorphism of a modular abelian variety A.

`RestrictEndomorphism(phi, A)`

The restriction of ϕ to an endomorphism of A, if this obviously makes sense. If A is not left invariant by ϕ, an error message may result.

`RestrictEndomorphism(phi, i)`

Suppose ϕ is an endomorphism of an abelian variety A and and $i : B \to A$ is an injective morphism of abelian varieties such that `i(B)` is invariant under ϕ. This intrinsic computes the endomorphism ψ of B induced by ϕ. If `i(B)` is not left invariant by ϕ, an error message may result.

`Restriction(phi, A)`

The restriction of ϕ to a morphism from A to the codomain of ϕ, if this obviously makes sense.

`RestrictionToImage(phi, i)`

Suppose $i : A \to D$ and $\phi : D \to B$ are morphisms of abelian varieties. This intrinsic computes the restriction of ϕ to the image of i. The resulting map is an endomorphism of the image of i.

`SurjectivePart(phi)`

Let $\phi : A \to B$ be a homomorphism. This intrinsic returns the *surjective* homomorphism $\pi : A \to \phi(A)$ induced by ϕ.

`UniversalPropertyOfCokernel(pi, f)`

Uses the universal property of the cokernel to find the unique morphism with a certain property. More precisely, suppose $\pi : B \to C$ is the cokernel of a morphism, so π is surjective with kernel K. Suppose $f : B \to D$ is a morphism whose kernel contains K. By definition of cokernel, there exists a unique morphism $\psi : C \to D$ such that $\pi * \psi = f$. This intrinsic returns ψ. If we only have that the identity component of $\ker(\pi)$ is contained in `ker(f)`, then ψ will only be a homomorphism in the category of abelian varieties up to isogeny, i.e., it will have a nontrivial denominator.

Example H0E27

We use the `Evaluate` and `Kernel` commands to cut out a 2-dimensional abelian subvariety of $J_0(65)$.

```
> J := Jzero(65);
> R<x> := PolynomialRing(RationalField());
> T2 := HeckeOperator(J,2);
> Factorization(CharacteristicPolynomial(T2));
[
    <x + 1, 2>,
    <x^2 - 3, 2>,
    <x^2 + 2*x - 1, 2>
]
> phi := Evaluate(x^2-3,T2);
> _,A := Kernel(phi);
> A;
Modular abelian variety of dimension 2 and level 5*13 over Q
```

The next example illustrates the universal property of the cokernel. We decompose $J = J_0(65)$ as a product of simples A, B, and C, of dimensions 1, 2, and 2, respectively. Then we let *pi* be a homomorphism from J onto $B + C$, and f a homomorphism from J onto B. The universal property supplies a morphism ψ from $B + C$ to B such that $\pi * \psi = f$ (i.e., $f(x) = \psi(\pi(x))$ for all x).

```
> J := Jzero(65);
> A,B,C := Explode(Decomposition(J));
> pi := NaturalMap(J,B+C);
> IsSurjective(pi);
true
> f := NaturalMap(J,B);
> psi := UniversalPropertyOfCokernel(pi,f); psi;
Homomorphism from modular abelian variety of dimension 4 to 65B
(up to isogeny) on integral homology by:
 (not printing 8x4 matrix)
> Matrix(psi);
[ 1/2    0    0 -1/2]
[ 1/2    0 -1/2  1/2]
[   0 -1/2  1/2    0]
[   0    0 -1/2    1]
[ 1/2 -1/2 -1/2  1/2]
[ 1/2    0 -1/2  1/2]
[   0  1/2    0    0]
[   0    0  1/2    0]
> pi*psi eq f;    // apply pi then psi is the same as applying f.
true
> Denominator(psi);
2
```

Since ψ has a denominator of 2, it is only a morphism in the category of abelian varieties up to isogeny. This is because only the connected component of the kernel of π is contained in the kernel of f.

```
> G := Kernel(pi); G;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 2, 2, 2, 2 ]
> H, K := Kernel(f);
> H;
{ 0 }:  finitely generated subgroup of abelian variety with
invariants []
> G subset K;
false
```

Next we illustrate dividing out by integers, by dividing the 10 out of $10T_3$ acting on $J_0(23)$. Note that this division occurs in the full endomorphism ring, not just the Hecke algebra.

```
> phi := 10*HeckeOperator(Jzero(23),3); phi;
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[-10 -20  20   0]
[  0 -30  20 -20]
[ 20 -40  30 -20]
[ 20 -20   0  10]
> DivideOutIntegers(phi);
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[-1 -2  2  0]
[ 0 -3  2 -2]
[ 2 -4  3 -2]
[ 2 -2  0  1]
10
```

Next we illustrate the restriction commands using factors of $J_0(65)$.

```
> J := Jzero(65);
> A := Decomposition(J)[2]; A;
Modular abelian variety 65B of dimension 2, level 5*13 and
conductor 5^2*13^2 over Q
> T := HeckeOperator(J,3);
> Factorization(CharacteristicPolynomial(T));
[
    <x + 2, 2>,
    <x^2 - 2*x - 2, 2>,
    <x^2 - 2, 2>
]
> Restriction(T,A);
Homomorphism from modular abelian variety of dimension 2 to
Jzero(65) given on integral homology by:
[-1  0  0  1  0 -1  0  0  0  0]
[-1  3 -1  1 -3 -1  1  2 -1 -2]
[-1  1 -1  3 -1 -1 -1  0  1 -2]
```

```
[-1  0  0  3  0 -1 -2  0  2 -2]
> TonA := RestrictEndomorphism(T,A); TonA;
Homomorphism from 65B to 65B given on integral homology by:
[-1  0  0  1]
[-1  3 -1  1]
[-1  1 -1  3]
[-1  0  0  3]
> Factorization(CharacteristicPolynomial(TonA));
[
    <x^2 - 2*x - 2, 2>
]
```

Finally, we illustrate the `SurjectivePart` command on the non-surjective morphism $T_3 + 2$ of $J_0(65)$.

```
> phi := T+2;
> IsSurjective(phi);
false
> pi := SurjectivePart(phi);
> IsSurjective(pi);
true
> Codomain(pi);
Modular abelian variety of dimension 4 and level 5*13 over Q
```

0.4.3 Kernels

The category of abelian varieties is not an abelian category because kernels need not exist in this category. More precisely, if ϕ is a homomorphism $A \to B$ of abelian varieties, then the kernel of ϕ is usually not an abelian variety because it is rarely connected. Instead, the kernel fits into an exact sequence

$$0 \to C \to \ker(\phi) \to G \to 0,$$

where C is an abelian variety and G is a finite group, both defined over the same field as ϕ.

The `ConnectedKernel` command returns C.

The `ComponentGroupOfKernel` command returns G as a subgroup of A/C.

The `Kernel` command returns a finite subgroup of A that maps to G, the abelian variety C, and a map from C into A.

Note that if $C \neq 0$ then the finite group that the kernel command returns *is not canonical*, since it is just some finite group that maps onto C via quotienting out by C. For the canonical component group, use the `ComponentGroupOfKernel` command, which gives a group defined over the same base field as ϕ, and whose main drawback is that the component group is not contained in A.

`ComponentGroupOfKernel(phi)`

Component group of $\ker(\phi)$.

`ConnectedKernel(phi)`

The connected component C of $\ker(\phi)$ and a morphism from C to the domain of ϕ.

`Kernel(phi)`

A finite subgroup G of $A = \mathit{Domain}(\phi)$, an abelian variety C such that $\ker(\phi)$ equals $f(C) + G$, and an injective map f from C to A. If $C = 0$, then G has the same field of definition as ϕ; otherwise, G is only known to be defined over the algebraic closure.

Example H0E28

The kernel of $T_2 - 3$ on $J_0(65)$ is an extension of an abelian surface by a finite group isomorphis to $(\mathbf{Z}/2\mathbf{Z})^4$.

```
> J := Jzero(65);
> T := HeckeOperator(J,2);
> phi := T^2-3;
> ConnectedKernel(phi);
Modular abelian variety of dimension 2 and level 5*13 over Q
Homomorphism from modular abelian variety of dimension 2 to
Jzero(65) given on integral homology by:
[ 1  0  0  0  0  1 -1  0  1 -1]
[ 0  1  0  0 -1  0  0  1  0 -1]
[ 0  0  1  0  0  0 -1  1  1 -1]
[ 0  0  0  1  0  0 -1  0  1 -1]
> Kernel(phi);
Finitely generated subgroup of abelian variety with
invariants [ 2, 2, 2, 2 ]
Modular abelian variety of dimension 2 and level 5*13 over Q
Homomorphism from modular abelian variety of dimension 2 to
Jzero(65) given on integral homology by:
[ 1  0  0  0  0  1 -1  0  1 -1]
[ 0  1  0  0 -1  0  0  1  0 -1]
[ 0  0  1  0  0  0 -1  1  1 -1]
[ 0  0  0  1  0  0 -1  0  1 -1]
> G := ComponentGroupOfKernel(phi); G;
Finitely generated subgroup of abelian variety with
invariants [ 2, 2, 2, 2 ]
> FieldOfDefinition(G);
Rational Field
```

Though G is defined over $\mathbf{Q}$, the ambient variety of G is the quotient of $J_0(65)$ by the two dimensional connected component of the kernel of $T_2 - 3$.

```
> AmbientVariety(G);
```

```
Modular abelian variety of dimension 3 and level 5*13 over Q
```

On the other hand, the group returned by the `Kernel` command is a subgroup of $J_0(65)$.

```
> H, C := Kernel(phi);
> H;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 2, 2 ]
> FieldOfDefinition(H);
Algebraically closed field with no variables
> AmbientVariety(H);
Modular abelian variety Jzero(65) of dimension 5 and level 5*13
over Q
```

0.4.4 Images

These commands compute the image of a modular abelian variety or a finite group under a homomorphism ϕ of modular abelian varieties. Using `@@`, one can also compute a group lifting a given group. Note that this lift is not canonical unless ϕ has finite kernel, in which case the `G@@phi` is the full inverse image of G.

`phi(A)`

The image of A under ϕ, if this makes sense, i.e., if A is the domain of ϕ, or A has dimension 0, or one of the embeddings of A has codomain equal to the domain of ϕ.

`phi(G)`

The image of G under ϕ, if this makes sense. If A is the ambient variety of G, then this makes sense if A is the domain of ϕ, or A has dimension 0, or one of the embeddings of A has codomain equal to the domain of ϕ.

`Image(phi)`

The image C of ϕ, which is a modular abelian subvariety contained in the codomain of ϕ, a morphism from C to the codomain of ϕ, and a surjective morphism from the domain of ϕ to C.

`G @@ phi`

A finite group who image under ϕ is equal to G, if possible. If ϕ has finite kernel, then this is the exact inverse image of G under ϕ. If not, then this is a group generated by a choice of torsion inverse image for each generator of G, which may not be canonical.

Example H0E29

The image of $J_0(37)$ under T_2 is an elliptic curve.

```
> J := Jzero(37);
> phi := HeckeOperator(J,2);
> phi(J);
Modular abelian variety of dimension 1 and level 37 over Q
> Image(phi);
Modular abelian variety of dimension 1 and level 37 over Q
Homomorphism from modular abelian variety of dimension 1 to
Jzero(37) given on integral homology by:
[ 1 -1  1  0]
[ 1 -1 -1  1]
Homomorphism from Jzero(37) to modular abelian variety of dimension
1 given on integral homology by:
[ 0 -1]
[ 1  0]
[-1  1]
[ 0  0]
```

The Hecke operator T_2 maps the 2-torsion subgroup of $J_0(37)$ maps onto the 2-torsion subgroup of the image of T_2.

```
> G := nTorsionSubgroup(J,2); G;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 2, 2 ]
> phi(G);
Finitely generated subgroup of abelian variety with invariants
[ 2, 2 ]
```

The homomorphism $T_2 - 1$ is surjective, and the inverse image of the 2-torsion is a subgroup isomorphic to $(\mathbf{Z}/2\mathbf{Z})^2 \times (\mathbf{Z}/6\mathbf{Z})^2$.

```
> IsSurjective(phi-1);
true
> psi := phi-1;
> H := G@@(psi); H;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 6, 6 ]
> psi(H);
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 2, 2 ]
> H := G@@psi; H;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 6, 6 ]
> psi(H);
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 2, 2 ]
```

0.4.5 Cokernels

`Cokernel(phi)`

The cokernel of ϕ and a morphism from the codomain of ϕ to the cokernel.

Example H0E30

We compute a 2-dimensional quotient of the 3-dimensional abelian variety $J_0(33)$ using the Hecke operator T_2 and the `Cokernel` command.

```
> J := Jzero(33);
> T := HeckeOperator(J,2);
> Factorization(CharacteristicPolynomial(T));
[
    <x - 1, 2>,
    <x + 2, 4>
]
> phi := T + 2;
> A, pi := Cokernel(phi);
> A;
Modular abelian variety of dimension 2 and level 3*11 over Q
> pi;
Homomorphism from Jzero(33) to modular abelian variety of dimension
2 (not printing 6x4 matrix)
```

0.4.6 Matrix structure

Homomorphisms are stored and worked with internally using the linear maps they induce on homology. The commands below provide access to those matrices.

`Eltseq(phi)`

The Eltseq of the underlying matrix that defines ϕ. This is a sequence of integers or rational numbers.

`IntegralMatrix(phi)`

The matrix which defines ϕ, written with respect to integral homology.

`IntegralMatrixOverQ(phi)`

The matrix which defines ϕ, written with respect to integral homology.

`Matrix(phi)`

The matrix on chosen basis of rational homology that defines ϕ.

`Ncols(phi)`

The number of columns of the matrix that defines ϕ. This is also the dimension of the homology of the codomain of ϕ.

`Nrows(phi)`

The number of rows of the matrix that defines ϕ. This is also the dimension of the homology of the domain of ϕ.

`RealMatrix(phi)`

The matrix which defines ϕ, written with respect to real homology.

`Rows(phi)`

The sequence rows of the matrix that defines ϕ.

Example H0E31

The following example demonstrates each of the commands for the Hecke operator T_2 on $J_0(23)$.

```
> phi := HeckeOperator(Jzero(23),2); phi;
Homomorphism T2 from Jzero(23) to Jzero(23) given on integral homology by:
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
> Eltseq(phi);
[ 0, 1, -1, 0, 0, 1, -1, 1, -1, 2, -2, 1, -1, 1, 0, -1 ]
> IntegralMatrix(phi);
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
> Parent($1);
Full RMatrixSpace of 4 by 4 matrices over Integer Ring
> IntegralMatrixOverQ(phi);
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
> Parent($1);
Full Matrix Algebra of degree 4 over Rational Field
> Matrix(phi);
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
> Ncols(phi);
4
> Nrows(phi);
4
```

```
> RealMatrix(phi);
[   0    1   -1    0]
[   0    1   -1    1]
[  -1  2/1 -2/1    1]
[  -1    1    0   -1]
> Parent($1);
Full KMatrixSpace of 4 by 4 matrices over Real Field
> Rows(phi);
[
    ( 0  1 -1  0),
    ( 0  1 -1  1),
    (-1  2 -2  1),
    (-1  1  0 -1)
]
```

0.4.7 Arithmetic

MAGMA supports many standard arithmetic operations with homomorphisms of abelian varieties, including composition, addition, subtraction, and exponentiation.

`psi(phi)`

The composition $\phi(\psi)$.

`Inverse(phi)`

The inverse of ϕ and an integer d such that $d * \phi^($ − 1) is a morphism of abelian varieties. More precisely, if ϕ is an isogeny, then the inverse of ϕ is a morphism in the category of abelian varieties up to isogeny. This intrinsic returns such a morphism or the actual inverse of ϕ if ϕ has degree 1.

`a * phi`

The product of the rational number a times the homomorphism ϕ. The result might only be a homomorphism, up to isogeny.

`a * phi`

The product of the integer a times the homomorphism ϕ.

`n + phi`

The sum of multiplication by the rational number n and the endomorphism ϕ.

`n + phi`

The sum of the endomorphism multiplication-by-n and the endomorphism ϕ.

`n - phi`

The difference of multiplication-by-n and the endomorphism ϕ.

`n - phi`

The difference of the endomorphism multiplication-by-n and the endomorphism ϕ.

```
phi * psi
```

The product of the matrix that defines ϕ and the matrix ψ.

```
phi * psi
```

The composition of the homomorphism ϕ and ψ.

```
phi * psi
```

The product of the matrix that defines ϕ and the matrix ψ.

```
phi + n
```

The sum of the endomorphism ϕ and the endomorphism multiplication-by-n.

```
phi + psi
```

The sum of the matrix that defines ϕ and the matrix ψ.

```
phi + psi
```

The sum of the homomorphism ϕ and ψ

```
phi + psi
```

The sum of the matrix that defines ϕ and the matrix ψ.

```
phi - n
```

The difference of the endomorphism ϕ and the endomorphism multiplication-by-n.

```
phi - n
```

The difference of the endomorphism ϕ and the endomorphism multiplication-by-n.

```
phi - psi
```

The difference of the matrix that defines ϕ and the matrix ψ.

```
phi - psi
```

The difference of the homomorphism ϕ and ψ

```
phi - psi
```

The difference of the matrix that defines ϕ and the matrix ψ.

```
phi ^ n
```

The n-fold composition ϕ^n of the endomorphism ϕ with itself. If $n = -1$, then this is the inverse of ϕ, in which case ϕ must be an isogeny or there is an error.

```
psi * phi
```

The product of the matrix ψ and the matrix that defines ϕ.

```
psi * phi
```

The product of the matrix ψ and the matrix that defines ϕ.

```
psi + phi
```

The sum of the matrix ψ and the matrix that defines ϕ.

`psi + phi`

The sum of the matrix ψ and the matrix that defines ϕ.

`psi - phi`

The difference of the matrix ψ and the matrix that defines ϕ.

`psi - phi`

The difference of the matrix ψ and the matrix that defines ϕ.

Example H0E32

We illustrate several arithmetic operations use the Hecke operators T_2 and T_3 on $J_0(23)$.

```
> J := Jzero(23);
> phi := HeckeOperator(J,2);
> psi := HeckeOperator(J,3);
> Matrix(phi)*Matrix(psi);
[-2  1 -1  0]
[ 0 -1 -1  1]
[-1  2 -4  1]
[-1  1  0 -3]
> phi*psi;
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[-2  1 -1  0]
[ 0 -1 -1  1]
[-1  2 -4  1]
[-1  1  0 -3]
> Inverse(phi);
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[ 1  1 -1  0]
[ 0  2 -1  1]
[-1  2 -1  1]
[-1  1  0  0]
1
> 1/3*phi;
Homomorphism from Jzero(23) to Jzero(23) (up to isogeny) on integral
homology by:
[   0  1/3 -1/3    0]
[   0  1/3 -1/3  1/3]
[-1/3  2/3 -2/3  1/3]
[-1/3  1/3    0 -1/3]
> 2+phi;
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[ 2  1 -1  0]
[ 0  3 -1  1]
[-1  2  0  1]
[-1  1  0  1]
> phi+psi;
```

```
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[-1 -1  1  0]
[ 0 -2  1 -1]
[ 1 -2  1 -1]
[ 1 -1  0  0]
> phi^4;
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[ 2 -3  3  0]
[ 0 -1  3 -3]
[ 3 -6  8 -3]
[ 3 -3  0  5]
1
> phi^(-4);
Homomorphism from Jzero(23) to Jzero(23) given on integral homology by:
[ 5  3 -3  0]
[ 0  8 -3  3]
[-3  6 -1  3]
[-3  3  0  2]
1
```

0.4.8 Polynomials

The polynomial commands compute the characteristic and minimal polynomials of endomorphisms of modular abelian varieties. Each command requires that the input homomorphism be a genuine homomorphism (not just a morphism up to isogeny). The same computation could be done by first extracting the matrix of the endomorphism and using the analogous command on the matrix. However, in some special cases there may be special techniques which can be used to do the computation more quickly, so there can be some advantage to using the commands below.

`CharacteristicPolynomial(phi)`

The characteristic polynomial of the endomorphism ϕ. Internally we can sometimes use extra information about ϕ (how it was constructed, Deligne bounds, e.g.), so calling this intrinsic may be faster than just asking for the characteristic polynomial of the matrix of ϕ.

`FactoredCharacteristicPolynomial(phi)`

The factorization of the characteristic polynomial of the endomorphism ϕ. Internally we can sometimes use extra information about ϕ (how it was constructed, Deligne bounds, e.g.), so calling this intrinsic may be faster than just asking for the factorization of the characteristic polynomial of the matrix of ϕ. Also, these factored polynomials are cached.

`MinimalPolynomial(phi)`

The minimal polynomial of the endomorphism ϕ. Internally we can sometimes use extra information about ϕ (how it was constructed, Deligne bounds, e.g.), so calling this intrinsic may be faster than just asking for the characteristic polynomial of the matrix of ϕ.

Example H0E33

This example illustrates each of the polynomial commands for the Hecke operator T_2 on $J_0(66)$.

```
> J := Jzero(66);
> phi := HeckeOperator(J,2);
> R<x> := PolynomialRing(RationalField());
> CharacteristicPolynomial(phi);
x^18 + 4*x^17 + 8*x^16 + 8*x^15 + 6*x^14 - 20*x^12 - 56*x^11 -
    55*x^10 - 4*x^9 + 36*x^8 + 48*x^7 + 104*x^6 + 128*x^5 -
    32*x^4 - 192*x^3 - 112*x^2 + 64*x + 64
> CharacteristicPolynomial(Matrix(phi));
x^18 + 4*x^17 + 8*x^16 + 8*x^15 + 6*x^14 - 20*x^12 - 56*x^11 -
    55*x^10 - 4*x^9 + 36*x^8 + 48*x^7 + 104*x^6 + 128*x^5 -
    32*x^4 - 192*x^3 - 112*x^2 + 64*x + 64
> FactoredCharacteristicPolynomial(phi);
[
    <x - 1, 4>,
    <x + 1, 2>,
    <x^2 - x + 2, 2>,
    <x^2 + 2*x + 2, 4>
]
> MinimalPolynomial(phi);
x^6 + x^5 + x^4 + x^3 + 2*x^2 - 2*x - 4
> Factorization(MinimalPolynomial(phi));
[
    <x - 1, 1>,
    <x + 1, 1>,
    <x^2 - x + 2, 1>,
    <x^2 + 2*x + 2, 1>
]
```

0.4.9 Invariants

The `Domain` and `Codomain` commands give the domain and codomain of a homomorphism. The `Degree` command computes the cardinality of the kernel of the homomorphism, or 0 if the cardinality is infinite. The denominator of a homomorphism ϕ up to isogeny is the smallest integer n, so that $n * \phi$ is a genuine homomorphism. The `FieldOfDefinition` of a homomorphism is *some* field over which the homomorphism is defined, but it is not guaranteed to be minimal. The `Rank` and `Nullity` of a homorphism are the dimension

of the image and kernel, respectively. The `Trace` of a homomorphism is the trace of any matrix that represents its action on integral homology, so e.g. the trace of multiplication by n on A is $2n\dim(A)$.

`ClearDenominator(phi)`

$Mor\phi sm$ $n * \phi$ with n positive and minimal.

`Codomain(phi)`

The codomain of ϕ.

`Degree(phi)`

The degree of the morphism ϕ. If the kernel of ϕ is not finite, then the degree is by definition 0.

`Denominator(phi)`

The smallest positive integer n such that $n * \phi$ is a homomorphism. This is also the denominator of the matrix that defines ϕ.

`Domain(phi)`

The domain of ϕ.

`FieldOfDefinition(phi)`

A field of definition of ϕ.

`Nullity(phi)`

The dimension of the kernel of ϕ.

`Rank(phi)`

The dimension of the kernel of ϕ.

`Trace(phi)`

The trace of a matrix that defines ϕ.

Example H0E34

```
> phi := NaturalMap(Jzero(11),Jzero(33));
> Codomain(phi);
Modular abelian variety Jzero(33) of dimension 3 and level 3*11 over Q
> Domain(phi);
Modular abelian variety Jzero(11) of dimension 1 and level 11 over Q
> Degree(phi);
1
> Denominator(1/5*phi);
5
> FieldOfDefinition(phi);
```

```
Rational Field
> Nullity(phi);
0
> Rank(phi);
1
> Trace(HeckeOperator(Jzero(33),2));
-6
> Trace(nIsogeny(Jzero(33),5));
30
```

0.4.10 Predicates

Suppose $\phi : A \to B$ is a homomorphism in the category of abelian varieties, possibly up to isogeny. The `IsMorphism` command returns true if ϕ is an actual morphism (i.e., no denominator), and the `OnlyUpToIsogeny` command returns true exactly when ϕ is not an actual morphism.

The `HasFiniteKernel` command return true exactly when ϕ has a finite kernel, and the `IsInjective` command returns true when the kernel of ϕ is 0. The `IsSurjective` command returns true if the image of ϕ is its codomain. The `IsIsogeny` command returns true if ϕ is surjective and has finite kernel. Note that just having finite kernel is not enough, so isogeny is an equivalence relation.

There are also commands for testing equality of homomorphisms and inclusion in a list.

`HasFiniteKernel(phi)`

True if the kernel of the homomorphism ϕ is finite.

`IsEndomorphism(phi)`

True if ϕ is an endomorphism, i.e., the domain and codomain of ϕ are equal.

`IsHeckeOperator(phi)`

If ϕ was computed using the HeckeOperator command, then true and the parameter n passed to the HeckeOperator command when creating ϕ. Otherwise false and 0.

`IsInjective(phi)`

True if ϕ is an injective homomorphism.

`IsInteger(phi)`

True if ϕ is multiplication by n for some integer n. If true, returns that n as well. If false, returns false and the second return value is not defined.

`IsIsogeny(phi)`

True if ϕ is a surjective homomorphism with finite kernel. Note that this definition differs from the one in Silverman's books on elliptic curves, agrees with the one in Milne's articles, and is an equivalence relation.

`IsIsomorphism(phi)`

True if ϕ is an isomorphism of abelian varieties.

`IsMorphism(phi)`

True if and only if ϕ is a morphism in the category of abelian varieties (not just in the category of abelian varieties up to isogeny).

`IsOptimal(phi)`

True if ϕ is an optimal quotient map, i.e., ϕ is surjective and has connected kernel.

`IsSurjective(phi)`

True if the homomorphism ϕ is surjective.

`IsZero(phi)`

True if ϕ is the zero morphism.

`OnlyUpToIsogeny(phi)`

True if ϕ is not a homomorphism, but $n * \phi$ is a homomorphism for some positive integer n, i.e., ϕ is only a homomorphism in the category of abelian varieties up to isogeny.

`n eq phi`

True if ϕ is equal to multiplication by the integer n.

`phi eq n`

True if ϕ is equal to multiplication by the integer n.

`phi eq psi`

True if the two homomorphisms ϕ and ψ are equal.

`phi in X`

True if ϕ is one of the homomorphisms in the list X of homomorphisms.

Example H0E35

```
> phi := HeckeOperator(Jzero(65),2)-1;
> HasFiniteKernel(phi);
true
> IsEndomorphism(phi);
true
> IsHeckeOperator(phi);
false 0
```

```
> IsInjective(phi);
false
> IsInteger(phi);
false
> IsIsogeny(phi);
true
> IsIsomorphism(phi);
false
> IsMorphism(phi);
true
> IsMorphism(1/2*phi);
false
> IsSurjective(phi);
true
> IsZero(phi);
false
> OnlyUpToIsogeny(phi);
false
> 2 eq phi;
false
> phi eq 2;
false
> 2 eq nIsogeny(Jzero(65),2);
true
> phi eq nIsogeny(Jzero(65),2);
false
> phi in [* phi, nIsogeny(Jzero(65),2) *];
true
> IsIsomorphism(NaturalMap(Jzero(11),Jone(11)));
false
> IsIsomorphism(NaturalMap(Jzero(11)^2,Jzero(22)));
false
> IsIsomorphism(NaturalMap(Jzero(11),Jzero(11)));
true
```

0.5 Endomorphism Algebras and Hom Spaces

0.5.1 Creation

Let A and B be modular abelian varieties. The `Hom` command creates the finite-rank free abelian group of homomorphisms from A to B, or with a third optional argument, the vector space of homomorphisms in the category of abelian varieties up to isogeny. The `End` command creates the endomorphism algebra of an abelian varieties. For the `Hom` and `End` commands no nontrivial computations are done by MAGMA until you ask for further information, such as the rank or a basis. In particular, creation of `Hom(A,B)` does not compute `Hom(A,B)`.

The `BaseExtend` command can alternatively be used to construct the the tensor product of $\mathrm{Hom}(A, B)$ with $\mathbf{Q}$. The only second arguments to the `BaseExtend` command are $\mathbf{Z}$ and $\mathbf{Q}$.

The `HeckeAlgebra` command defines a commutative subring of $\mathrm{End}(A)$ generated by the Hecke operators. This is defined for any modular abelian variety A, possibly using pullbacks and projections from an abelian variety attached to modular symbols (which in some cases can produce a Hecke algebra that differs from what you might expect by some finite index).

`BaseExtend(H, R)`

The space $H \otimes R$, where R is the rational numbers or integers. When $R = \mathbf{Q}$, this is the space of homomorphisms in the category of abelian varieties up to isogeny.

`End(A)`

The endomorphism ring of A.

`End(A, overQ)`

If overQ is false, the endomorphism ring, and if it is true, the endomorphism algebra.

`HeckeAlgebra(A)`

The Hecke algebra associated to A. For an abelian variety attached to modular symbols, this is the algebra induced by Hecke operators acting on modular symbols (homology). For a general abelian variety, let $\pi : J \to A$ and $e : A \to J$ be the modular parameterization and embedding of A. Then this is the ring of endomorphisms obtained by pulling back the Hecke algebra on J to A using π and e, i.e., it is $e * T * \pi$, where T is the Hecke ring of J. For example, since $J_1(N)$ is represented as a quotient of $J_0(N)$, the Hecke algebra is not what you might expect.

`Hom(A, B)`

Group of homomorphisms from A to B.

`Hom(A, B, overQ)`

Group of homomorphisms from A to B, or vector space generated by homomorphisms from A to B if overQ is true.

Example H0E36

```
> A := Jzero(11); B := Jzero(33);
> Hom(A,B);
Group of homomorphisms from Jzero(11) to Jzero(33)
> Hom(A,B,true);
Group of homomorphisms from Jzero(11) to Jzero(33) in the category of
abelian varieties up to isogeny
> End(A);
Group of homomorphisms from Jzero(11) to Jzero(11)
> End(A,true);
Group of homomorphisms from Jzero(11) to Jzero(11) in the category of
abelian varieties up to isogeny
> BaseExtend(Hom(A,B),RationalField());
_Q: Group of homomorphisms from Jzero(11) to Jzero(33) in the category
of abelian varieties up to isogeny
> HeckeAlgebra(A);
HeckeAlg(Jzero(11)):  Group of homomorphisms from Jzero(11) to Jzero(11)
```

0.5.2 Subgroups and subrings

The `Saturation` command computes the saturation of a subgroup H of $\mathrm{Hom}(A, B)$ in the full $\mathrm{Hom}(A, B)$. This is a subgroup S of $\mathrm{Hom}(A, B)$ such that S contains H with finite index and the quotient $\mathrm{Hom}(A, B)/S$ is torsion free.

Use the `Subgroup` command to construct the subgroup of $\mathrm{Hom}(A, B)$ generated by certain elements.

The `Subring` command constructs the subring generated by an element of $\mathrm{End}(A)$. Note that a ring does not have to contain unity.

`RingGeneratedBy(H)`

The ring of endomorphisms generated by the endomorphisms in H.

`Saturation(H)`

The saturation of H in all homomorphisms. Suppose A and B are abelian varieties and H is a subgroup of `Hom(A,B)`. Then `Hom(A,B)` is a free $\mathbf{Z}$-module, and the saturation of H in `Hom(A,B)` is a group H' that contains H with finite index such that the quotient of `Hom(A,B)` by H' is torsion free.

`Subgroup(X)`

Group of homomorphisms from A to B generated by elements of X.

`Subgroup(X, overQ :` *parameters*`)`

`IsBasis`	BoolElt	*Default :* $false$

Group of homomorphisms generated by elements of X. If the parameter IsBasis is true, then we assume the elements of X are a basis for their span.

`Subring(X)`

Group of homomorphisms from A to B generated by elements of X.

`Subring(X, overQ)`

The ring of endomorphisms generated by elements in X.

`Subring(phi)`

The ring of endomorphisms generated by elements in ϕ.

Example H0E37

In this example we use the `Saturation` command to find the integers N up to 60 such that the Hecke algebra of $J_0(N)$ is not saturated in the full ring of endomorphisms.

```
> function ind(N)
>    H := HeckeAlgebra(Jzero(N));
>    return Index(Saturation(H),H);
> end function;
> for N in [2..60] do
>    i := ind(N);
>    if i gt 1 then print N, i; end if;
> end for;
44 2
46 2
54 3
56 2
60 2
```

Note that multiplicity one fails at 3 for $J_0(54)$. It might be interesting to find a precise relationship between failure of multiplicity one and the index of the Hecke algebra T in its saturation.
The next example illustrates constructing a subgroup.

```
> J := Jzero(33);
> E := End(J); E;
Group of homomorphisms from Jzero(33) to Jzero(33)
> H := Subgroup([E.1, E.3]); H;
Group of homomorphisms from Jzero(33) to Jzero(33)
> Rank(H);
2
```

We compute the subring generated by T_2 on $J_0(100)$.

```
> T2 := HeckeOperator(Jzero(100),2);
> R := Subring(T2); R;
Group of homomorphisms from Jzero(100) to Jzero(100)
> Rank(R);
3
```

The Hecke operator T_2 and the main Atkin-Lehner involution together generate a commutative ring of rank 10 over $\mathbf{Z}$.

```
> J := Jzero(100);
> T2 := HeckeOperator(J,2);W := AtkinLehnerOperator(J,100);
> R := Subring([End(J) | T2,W]);
> Dimension(R);
10
> Dimension(End(J));
13
> IsRing(R);
true
> IsCommutative(R);
false
```

0.5.3 Pullback and pushforward of Hom spaces

A homomorphism of abelian varieties induces a map from one space of homomorphisms into another, and the three commands below compute the image of such maps.

`Pullback(H, phi)`

Given a space of homomorphism H in `Hom(A,B)` and a morphism $\phi : B \to C$, compute the image of H in `Hom(A,C)` via the map that sends f to $f * \phi$.

`Pullback(phi, H)`

Given a space of homomorphism H in `Hom(A,B)` and a morphism $\phi : C \to A$, compute the image of H in `Hom(C,B)` via the map that sends f to $\phi * f$.

`Pullback(phi, H, psi)`

Suppose H is a space of homomorphism $A \to B$ and $\phi : C \to A$ and $\psi : B \to D$. Then this intrinsic computes and returns the ring of homomorphisms of A of the form $\phi * f * \psi$, where f is in H.

Example H0E38

```
> H := Hom(Jzero(11),Jzero(22));
> phi := NaturalMap(Jzero(22),Jzero(33));
> psi := NaturalMap(Jzero(33),Jzero(11));
> Pullback(H,phi);
Group of homomorphisms from Jzero(11) to Jzero(33)
> Pullback(psi,H);
Group of homomorphisms from Jzero(33) to Jzero(22)
> Pullback(psi,H,phi);
Group of homomorphisms from Jzero(33) to Jzero(33)
```

0.5.4 Arithmetic

The + and `meet` command compute the sum and intersection of two subgroups of $\mathrm{Hom}(A, B)$.

`H1 + H2`

The subgroup generated by H_1 and H_2, where H_1 and H_2 are assumed to both be subgroups of `Hom(A,B)`, for abelian varieties A and B.

`H1 meet H2`

The intersection of H_1 and H_2, where H_1 and H_2 are assumed to both be subgroups of `Hom(A,B)`, for abelian varieties A and B.

Example H0E39

```
> J := Jzero(33);
> E := End(J);
> H1 := HeckeAlgebra(J); H1;
HeckeAlg(Jzero(33)):  Group of homomorphisms from Jzero(33) to Jzero(33)
> H2 := Subgroup([E.1,E.2]); H2;
Group of homomorphisms from Jzero(33) to Jzero(33)
> Dimension(E);
5
> Dimension(H1);
3
> Dimension(H2);
2
> Dimension(H1 meet H2);
1
> Dimension(H1 + H2);
4
```

0.5.5 Quotients

If H_1 and H_2 are subspace of $\mathrm{Hom}(A, B)$ with H_1 contained in H_2, then the `Index` command compute the index of H_1 in H_2. If H_1 is not contained in H_2 then the generalized lattice index is computed instead. The `Quotient` command computes the quotient H_2/H_1 and various natural maps.

`Index(H2, H1)`

The index of H_1 in H_2, where H_1 and H_2 are both subgroups of `Hom(A,B)`, for abelian varieties A and B. If H_1 is contained in H_2, this is just the cardinality of H_2/H_1, or 0 if this cardinality is infinite. If H_1 is not contained in H_2, then the index is by definition $[H_1 + H_2 : H_1]/[H_1 + H_2 : H_2]$, assuming the denominator is

nonzero, i.e., that H_2 has finite index in $H_1 + H_2$ (it is an error if H_2 does not have finite index in $H_1 + H_2$).

`Quotient(H2, H1)`

The abelian group quotient H_2/H_1, a map from H_2 to this quotient, and a lifting map from this quotient to H_2.

`H2 / H1`

The abelian group quotient H_2/H_1, a map from H_2 to this quotient, and a lifting map from this quotient to H_2.

Example H0E40

We define the subgroup of $\mathrm{End}(J_0(54))$ generated by T_1, T_2, T_3, and T_4, find that it has infinite index in the full Hecke algebra, and compute the quotient, which is $\mathbf{Z}$.

```
> J := Jzero(54);
> T := HeckeAlgebra(J);
> Dimension(T);
4
> S := Subgroup([HeckeOperator(J,n) :  n in [1..4]]);
> Dimension(S);
3
> Index(T,S);
0
> Quotient(T,S);
Abelian Group isomorphic to Z
Defined on 1 generator (free)
Mapping from:  HomModAbVar:  T to Abelian Group isomorphic to Z
Defined on 1 generator (free) given by a rule [no inverse]
Mapping from:  Abelian Group isomorphic to Z
Defined on 1 generator (free) to HomModAbVar:  T given by a rule
[no inverse]
> G := T/S; G;
Abelian Group isomorphic to Z
Defined on 1 generator (free)
```

We compute the subgroup generated by T_3, T_4, T_5, and T_{10}, and find that it has index 6 in its saturation.

```
> S := Subgroup([HeckeOperator(J,n) :  n in [3,4,5,10]]);
> Sat := Saturation(S);
> Index(Sat,S);
6
> Index(S,Sat);
1/6
> Invariants(Sat/S);
[ 6 ]
```

0.5.6 Invariants

The `Domain` and `Codomain` command give the domain and codomain of the elements of a space of homomorphisms. The `FieldOfDefinition` is some field that all the homomorphisms in the Hom space are defined over, but it is not guaranteed to be minimal.

The `Discriminant` command computes the discriminant of the trace pairing on a Hom space. Note that the trace is the trace of the action on homology. Computation of the discriminant can be time consuming. Discriminants of Hecke algebras are particularly interesting to compute because they are closely related to congruences between eigenforms.

`Codomain(H)`

The codomain of the homomorphisms in H.

`Discriminant(H)`

The discriminant of H with respect to the trace pairing matrix. This is the trace of endomorphisms acting on homology, not left multiplication on themselves, so, e.g., the discriminant of the Hecke algebra will be 2^d times as big as it would be otherwise (if the sign is 0), where d is the dimension of A. If H is over $\mathbf{Q}$, this function returns the discriminant of the lattice of elements in H that are homomorphisms.

`Domain(H)`

The domain of the homomorphisms in H.

`FieldOfDefinition(H)`

A field over which all homomorphisms in H are defined.

Example H0E41

```
> H := Hom(Jzero(11),Jzero(33));
> Domain(H);
Modular abelian variety Jzero(11) of dimension 1 and level 11 over Q
> Codomain(H);
Modular abelian variety Jzero(33) of dimension 3 and level 3*11 over
Q
> FieldOfDefinition(H);
Rational Field
> A := BaseExtend(Jzero(11),ComplexField());
> H := End(A);
> FieldOfDefinition(H);
Complex Field
```

Though $H = \mathbf{Z}$, so the discriminant of the abstract ring H is 1, the discriminant of the trace pairing of H acting on homology is 2:

```
> Discriminant(H);
2
[2]
```

The prime $p = 389$ is the only known examples where p divides the discriminant of the Hecke algebra of $J_0(p)$.

```
> T := HeckeAlgebra(Jzero(389,2,+1));
> d := Discriminant(T);
> J := Jzero(389,2,+1);
> T := HeckeAlgebra(J);
> d := Discriminant(T);
> d mod 389;
0
> Factorization(d);
[ <2, 53>, <3, 4>, <5, 6>, <31, 2>, <37, 1>, <389, 1>, <3881, 1>,
<215517113148241, 1>, <477439237737571441, 1> ]
```

All the "action" at 389 comes from the 20-dimensional simple factor.

```
> A := J(5); A;
Modular abelian variety 389E of dimension 20, level 389 and
conductor 389^20 over Q with sign 1
> Factorization(Discriminant(HeckeAlgebra(A)));
[ <2, 98>, <5, 41>, <389, 1>, <215517113148241, 1>,
<477439237737571441, 1> ]
```

0.5.7 Structural invariants

The following commands provide access to a basis and generators for spaces of homomorphisms.

`Basis(H)`

A basis for H.

`Dimension(H)`

The rank of H as a $\mathbf{Z}$-module or $\mathbf{Q}$-vector space.

`Generators(H)`

A basis for H.

`Ngens(H)`

The number of generators of H.

`Rank(H)`

The rank of H as a $\mathbf{Z}$-module or $\mathbf{Q}$-vector space.

`H . i`

The ith generator of H.

Example H0E42

The following example illustrates each of the commands for $\mathrm{Hom}(J_0(11), J_0(33))$.

```
> H := Hom(Jzero(11),Jzero(33));
> Basis(H);
[
    Homomorphism from Jzero(11) to Jzero(33) given on integral homology
    by:
    [ 0  2 -1 -2  0  1]
    [ 1  0 -1 -1  1  1],
    Homomorphism from Jzero(11) to Jzero(33) given on integral homology
    by:
    [ 1  0 -2  2 -3  0]
    [ 1 -1  0  1 -2  1]
]
> Dimension(H);
2
> Ngens(H);
2
> Rank(H);
2
> H.1;
Homomorphism from Jzero(11) to Jzero(33) given on integral homology by:
[ 0  2 -1 -2  0  1]
[ 1  0 -1 -1  1  1]
```

0.5.8 Matrix and module structure

The following commands associate lattices, vector spaces, matrix algebras, and matrix spaces to subspaces H of $\mathrm{Hom}(A, B)$. The `Lattice` command takes a basis for H, `Eltseq`'s each element, and construct a free **Z**-module out of the result. The `MatrixAlgebra` command creates the algebra *generated by* the matrices of generators for H. The `RModuleWithAction` command creates a module over the ring R generated by H, and this module is equipped with an action of R.

`Lattice(H)`

A lattice with basis obtained from the components of the matrices of a basis for H.

`MatrixAlgebra(H)`

The matrix algebra generated by the underlying matrices of all elements in H, acting on homology.

`RMatrixSpace(H)`

Matrix space whose basis are the generators for H.

`RModuleWithAction(H)`

A module over H equipped with the action of H, where H must be a ring of endomorphism.

`RModuleWithAction(H, p)`

A module over H tensor F_p equipped with the action of H tensor F_p, where H must be a ring of endomorphism that has not been tensored with $\mathbf{Q}$.

`VectorSpace(H)`

A vector space with basis obtained from the components of the matrices of a basis for H.

Example H0E43

We first demonstrate some of the commands with $\mathrm{Hom}(J_0(11), J_0(33))$.

```
> H := Hom(Jzero(11),Jzero(33));
> Lattice(H);
Lattice of rank 2 and degree 12
Basis:
( 0  2 -1 -2  0  1  1  0 -1 -1  1  1)
( 1  0 -2  2 -3  0  1 -1  0  1 -2  1)
> RMatrixSpace(H);
RMatrixSpace of 2 by 6 matrices and dimension 2 over Integer Ring
> RMatrixSpace(BaseExtend(H,RationalField()));
KMatrixSpace of 2 by 6 matrices and dimension 2 over Rational
Field
> VectorSpace(H);
Vector space of degree 12, dimension 2 over Rational Field
User basis:
( 1  0 -2  2 -3  0  1 -1  0  1 -2  1)
( 0 -2  1  2  0 -1 -1  0  1  1 -1 -1)
```

Next we consider the endomorphism algebra of $J_0(22)$.

```
> H := End(Jzero(22));
> MatrixAlgebra(H);
Matrix Algebra of degree 4 with 4 generators over Integer Ring
> RModuleWithAction(H);
RModule(IntegerRing(), 4)
Action:
[1 0 0 0]
[0 1 0 0]
[0 0 1 0]
[0 0 0 1]
[ 0  1  0  1]
[ 0  0  0  0]
[ 1  0  1 -1]
```

```
[ 0  1  0  1]
[ 0  1  0 -1]
[ 0  1  0  0]
[-1  2 -1  1]
[-1  1  0  0]
[ 0  1 -2  1]
[-1  2 -1  0]
[-1  0  1 -1]
[ 0 -1  1 -1]
```

The following example illustrates that `MatrixAlgebra` computes the algebra generated by H, even if H is not itself an algebra.

```
> H := Subgroup([HeckeOperator(Jzero(33),2)]); H;
Group of homomorphisms from Jzero(33) to Jzero(33)
> A := MatrixAlgebra(H); A;
Matrix Algebra of degree 6 with 1 generator over Integer Ring
> Dimension(A);
2
> Dimension(H);
1
```

0.5.9 Predicates

These commands allow you to determine whether a space of homomorphisms is a ring, if it's commutative, if it's a field (and what field), if it was created using the `HeckeAlgebra` command, whether it has been tensored with $\mathbf{Q}$, or whether it is satured in the full ring of endomorphisms. You can also test equality and inclusion.

`IsCommutative(H)`

True if and only if H is a commutative ring.

`IsField(H)`

True if H is a field, and if so returns that field, a map from the field to H, and a map from H to the field.

`IsHeckeAlgebra(H)`

True if H was constructed using the HeckeAlgebra command. If H was constructed in another way, you should use the HeckeAlgebra command to compute the Hecke algebra and compare it with H.

`IsOverQ(H)`

True if H is a $\mathbf{Q}$-vector space instead of just a $\mathbf{Z}$-module, i.e., a space of homomorphisms up to isogeny.

`IsRing(H)`

True if H is a ring. (Note that a ring does not have to contain unity.)

`IsSaturated(H)`

True if H is equal to its saturation, i.e., the quotient of the ambient `Hom(A,B)` by H is torsion free.

`H1 eq H2`

True if H_1 and H_2 are equal.

`H1 subset H2`

True if H_1 and H_2 are both subgroups of a common `Hom(A,B)`, and in addition H_1 is a subset of H_2.

Example H0E44

We illustrate several of the commands for the endomorphism ring of $J_0(33)$.

```
> H := End(Jzero(33));
> IsCommutative(H);
false
> IsField(H);
false 0 0 0
> IsHeckeAlgebra(H);
false
> IsOverQ(H);
false
> IsOverQ(BaseExtend(H,RationalField()));
true
> IsRing(H);
true
> IsSaturated(H);
true
```

Next we compare the endomorphism ring with the Hecke algebra of $J_0(33)$.

```
> T := HeckeAlgebra(Jzero(33));
> T eq H;
false
> T subset H;
true
> IsSaturated(T);
true
> IsRing(T);
true
> IsHeckeAlgebra(T);
true
> IsCommutative(T);
true
```

```
> IsField(BaseExtend(T,RationalField()));
false 0 0 0
```

The Hecke algebra of $J_0(33)$ is actually a product of 3 fields, so it is not a field. In contract, the Hecke algebra of $J_0(23)$ is a field.

```
> T := HeckeAlgebra(Jzero(23));
> IsField(T);
false 0 0 0
```

In the following code, the answer you get might be different, since computation of the defining polynomial for the number field involves a randomized algorithm.

```
> IsField(BaseExtend(T,RationalField()));
true Number Field with defining polynomial x^2 + 11*x + 29 over
the Rational Field
Mapping from:  Number Field with defining polynomial x^2 + 11*x +
    29 over the Rational Field to HeckeAlg(Jzero(23))_Q: Group of
homomorphisms from Jzero(23) to Jzero(23) in the category of abelian
varieties up to isogeny  given by a rule [no inverse]
Mapping from:  HeckeAlg(Jzero(23))_Q: Group of homomorphisms from
Jzero(23) to Jzero(23) in the category of abelian varieties up to
isogeny  to Number Field with defining polynomial x^2 + 11*x + 29
over the Rational Field given by a rule [no inverse]
```

0.5.10 Random element

This command provides a random element of a homspace of rank r. It uses the MAGMA `Random` command to compute a random element of `RSpace(ZZ,r)`.

`Random(H)`

A random element of H.

`H ! x`

Coerce x into H.

Example H0E45

```
> H := End(Jzero(22));
> Random(H);
Homomorphism from Jzero(22) to Jzero(22) given on integral homology by:
[ 9 -4  0  2]
[ 0  6  0  0]
[ 2 -6 11 -2]
[ 3 -4  0  8]
```

0.6 Artihmetic of Abelian Varieties

0.6.1 Direct sum

One can take arbitrary finite direct sums of modular abelian varieties using the `DirectSum` command. The notation `A*B` is a shorthand for the direct sum of A and B. We do not write `A+B` for the direct sum, since it is already used for the sum of A and B inside a common ambient abelian variety, and this sum need not be direct, unless $A \cap B = 0$.

`DirectProduct(A, B)`

Same as DirectSum.

`DirectProduct(X)`

Same as DirectSum.

`DirectSum(A, B)`

The direct sum D of A and B, together with the embedding maps from A into D and B into D, respectively, and the projection maps from D onto A and B, respectively.

`DirectSum(X)`

The direct sum D of the sequence X of modular abelian varieties, together with a list containing the embedding maps from each modular abelian variety of X into D and a list containing the projection maps from D onto each modular abelian variety in X.

`A * B`

The direct sum of A and B.

`A ^ n`

The direct sum of n copies of A. If $n = 0$, the zero subvariety of A. If n is negative, the (-n)-th power of the dual of A.

0.6.2 Sum in an ambient variety

The sum `A+B` is the sum of A and B inside a common ambient abelian variety, and this sum need not be direct, unless the intersection of A and B is 0.

`FindCommonEmbeddings(X)`

True and a list of embeddings into a common abelian variety, if one can be found using `Embeddings(A)` for A in X. If true, the second return value is the list of embeddings.

`SumOf(X)`

The sum of the modular abelan varieties in X.

`SumOfImages(phi, psi)`

The sum D of the images of the morphisms ϕ and ψ in their common codomain, a morphism from D into their common codomain, and a list containing a morphism

from the domain of each of ϕ and ψ to D. If the codomains are not the same, then the homomorphisms are replaced by homomorphisms into an appropriate direct sum of codomains.

`SumOfMorphismImages(X)`

The sum D of the images of the morphisms in the list X of homomorphisms of modular abelian varieties in their common codomain, a morphism from D into their common codomain, and a list containing a morphism from the domain of each morphism in X to D. If not all codomains of the elements of X are the same, then the homomorphisms are replaced by homomorphisms into an appropriate direct sum of codomains.

`A + B`

The sum of the images of A and B in a common ambient abelian variety.

0.6.3 Intersections

Two abelian varieties cannot, by themselves, by intersected without choosing an embedding of both varieties in a common ambient abelian variety. The algorithm for computing an intersection is to compute the kernel of a certain homomorphism.

Intersections are computed in MAGMA by finding a homomorphism whose kernel is isomorphic to the intersection. For example, if $f : A \to C$ and $g : B \to C$ are injective homomorphisms, then the intersection of their images is isomorphic to the kernel of $f - g$.

As mentioned above, kernels of morphisms of abelian varieties are frequently not themselves abelian varieties. Instead a kernel is an extensions of an abelian variety by a finite group of components. Likewise, intersections of abelian varieties are often not abelian varieties.

The `ComponentGroupOfIntersection` command computes the group of components of the intersection of two abelian varieties (for more details, see the discussion of kernels in this section). The `Intersection` command computes a finite lift G of the group of components and an abelian variety C such that the relevant intersection is $C + G$.

The intersection commands also take a sequence of abelian varieties or list of morphisms in order to facilitate computation of n-fold intersections, for any positive integer n.

`ComponentGroupOfIntersection(A, B)`

Finite component group of intersection.

`ComponentGroupOfIntersection(X)`

Finite component group of intersection.

`Intersection(X)`

A finite lift of the component group of the intersection, the *connected* component of the intersection, and a map from the abelian variety that contains the connected component to the abelian variety that contains the component group. The elements of X are abelian varieties; they are replaced by their images via their modular

embedding map. All the elements of X must be embedded in the same abelian variety.

`IntersectionOfImages(X)`

A finite lift of the component group of the intersection, the *connected* component of the intersection, and a map from the abelian variety that contains the connected component to the abelian variety that contains the component group. The elements of X are morphisms from various abelian varieties into a common abelian variety. They do not have to be injective.

`A meet B`

The intersection of the abelian varieties A and B, in some natural ambient abelian variety. For more details, see the documentation for the Intersection command.

Example H0E46

The intersection of the three simple newform abelian subvarieties of $J_0(65)$ is a group isomorphic to $\mathbf{Z}/2\mathbf{Z}$.

```
> D := Decomposition(Jzero(65));
> G := ComponentGroupOfIntersection(D); G;
Finitely generated subgroup of abelian variety with
invariants [ 2 ]
> FieldOfDefinition(G);
Rational Field
```

The quotient of D[1] by this subgroup of order 2 is an elliptic curve over $\mathbf{Q}$ isogenous to D[1], but not isomorphic to D[1].

```
> B  := D[1]/G; B;
Modular abelian variety of dimension 1 and level 5*13 over Q
> IsIsomorphic(D[1],B);
false
```

Next we compute some non-finite intersections.

```
> A := D[1] + D[2];
> B := D[1] + D[3];
> A meet B;
Finitely generated subgroup of abelian variety with invariants [ 2, 2, 2 ]
Modular abelian variety of dimension 1 and level 5*13 over Q
Homomorphism from modular abelian variety of dimension 1 to
modular abelian variety of dimension 6 given on integral homology
by:
[ 1 -1  0  0  0  0  1 -1  0  0  0 -1]
[ 0  0  1 -1  1 -1  0  0  1 -1  1  0]
Homomorphism from modular abelian variety of dimension 6 to
Jzero(65) (not printing 12x10 matrix)
```

We can also intersect images of morphisms.

```
> f := ModularEmbedding(A);
```

```
> g := ModularEmbedding(B);
> _, C := IntersectionOfImages([* f, g *]);
> C eq D[1];
true
```

The following example illustrates failure of multiplicity one for $J_0(p)$ for a prime number p. This is the first such example known, and it was discovered by Lloyd Kilford using MAGMA (TODO: add reference to JNT paper). There are two elliptic curve factors A and B inside $J_0(431)$. The eigenforms associated to A and B are congruent modulo 2, but the intersection of A and B is trivial.

```
> J := Jzero(431);
> IsPrime(431);
true
> A := Decomposition(J)[1];
> B := Decomposition(J)[2];
> G, C := A meet B;
> G;
{ 0 }:  finitely generated subgroup of abelian variety with
invariants []
> C;
Modular abelian variety ZERO of dimension 0 and level 431 over Q
> Newform(A) - Newform(B);
-2*q^3 + 4*q^5 + 2*q^6 - 4*q^7 + O(q^8)
```

0.6.4 Quotients

If B is an abelian subvariety of A (or some natural image of B lies in A), then the quotient A/B is an abelian variety. Also, the cokernel of a homomorphism of abelian varieties is an abelian variety.

`Cokernel(phi)`

The cokernel of ϕ and a morphism from the codomain of ϕ to the cokernel.

`A / B`

The quotient of A by a natural image B' of B, if possible. Here B' is the image of B under the modular embedding composed with the modular parameterization to A.

Example H0E47

We compute a 2-dimensional quotient of the 3-dimensional abelian variety $J_0(33)$ using the Hecke operator T_2.

```
> J := Jzero(33);
> T := HeckeOperator(J,2);
> Factorization(CharacteristicPolynomial(T));
```

```
[
    <x - 1, 2>,
    <x + 2, 4>
]
> C := ConnectedKernel(T-1);
> B,psi := J/C;
> B;
Modular abelian variety of dimension 2 and level 3*11 over Q
```

0.7 Decomposing and Factoring Abelian Varieties

By the Poincare reducibility theorem, every abelian variety is isogenous to a product of simple abelian subvarieties. If A is a modular abelian variety over $\mathbf{Q}$, then A is isogenous to a product of simple abelian varieties A_f attached to newforms. The `Decomposition` and `Factorization` commands compute such decompositions.

0.7.1 Decomposition

Given an abelian variety A, the `Decomposition` command returns a sequence B_i of modular abelian varieties, such that their product is isogenous to A. Each B_i is equipped with an embedding into A such that the sum of the images of the B_i is equal to A. This embedding is the first element of the output of the `Embeddings` command.

`A(n)`

The nth factor in `Decomposition(A)`, denoted `A(n)`.

`Decomposition(A)`

Isogeny simple decomposition of A, i.e., a sequence of isogeny simple abelian subvarieties of A whose product is isogenous to A. Each is equipped with an embedding into A.

Example H0E48

We decomposition $A = J_0(37) \times J_0(22)$, then find the embedding into A of one of the factors that is isogenous to $J_0(11)$.

```
> A := Jzero(37) * Jzero(22);
> D := Decomposition(A); D;
[
    Modular abelian variety 37A of dimension 1, level 2*11*37 and
    conductor 37 over Q,
    Modular abelian variety 37B of dimension 1, level 2*11*37 and
    conductor 37 over Q,
    Modular abelian variety N(11,814,1)(11A) of dimension 1,
    level 2*11*37 and conductor 11 over Q,
```

```
    Modular abelian variety N(11,814,2)(11A) of dimension 1,
    level 2*11*37 and conductor 11 over Q
]
> B := D[3];
> Embeddings(B);
[*
Homomorphism from N(11,814,1)(11A) to Jzero(37) x Jzero(22) given on
integral homology by:
[ 0  0  0  0  1  0 -1  3]
[ 0  0  0  0  0  1 -2  3]
*]
```

0.7.2 Factorization

Given an abelian variety A, the `Factorization` command finds pairwise non-isogenous simple newform abelian varieties whose product, with multiplicities, is isomorphic to A.

`Factorisation(A)`

Synonym for Factorization.

`Factorization(A)`

Factorization of A as a product with multiplicities of abelian varieties $B = A_f$ attached to newforms. This is a list of pairs $< B, [\phi, ...] >$, where B is an isogeny simple abelian variety and $[\phi, ...]$ is a sequence of maps from B into A, such that the product of all images of all B is isogenous to A, and the sum of the dimensions of the images of B is the dimension of A. Moreover, the B are pairwise non-isogenous and are attached to newforms. To obtain a list of the images of the B canonically embeded into A, use `Decomposition(A)`.

Example H0E49

```
> A := Jzero(37) * Jzero(22);
> Factorization(A);
[*
<Modular abelian variety 37A of dimension 1, level 37 and
conductor 37 over Q, [
    Homomorphism N(37,814,1) from 37A to Jzero(37) x Jzero(22) given on
    integral homology by:
    [ 1 -1  1  0  0  0  0  0]
    [ 1 -1 -1  1  0  0  0  0]
]>,
<Modular abelian variety 37B of dimension 1, level 37 and
conductor 37 over Q, [
```

```
    Homomorphism N(37,814,1) from 37B to Jzero(37) x Jzero(22) given on
    integral homology by:
    [1 1 1 0 0 0 0 0]
    [0 0 0 1 0 0 0 0]
]>,
<Modular abelian variety 11A of dimension 1, level 11 and
conductor 11 over Q, [
    Homomorphism N(11,814,1) from 11A to Jzero(37) x Jzero(22) given on
    integral homology by:
    [ 0  0  0  0  0  1 -2  3]
    [ 0  0  0  0  1 -1  1  0],
    Homomorphism N(11,814,2) from 11A to Jzero(37) x Jzero(22) given on
    integral homology by:
    [ 0  0  0  0 -1  0  2 -2]
    [ 0  0  0  0 -1  2 -1  0]
]>
*]
```

0.7.3 Decomposition with respect to an endomorphism or a commutative ring

The following commands use the elements of a commutative subring of endomorphisms to decompose a modular abelian variety A into a direct sum of abelian subvarieties by taking kernels (which are analogous to generalized eigenspaces).

`DecomposeUsing(R)`

Decompose A using the commutative ring of endomorphisms generated by A.

`DecomposeUsing(phi)`

Decompose A using the endomorphism ϕ.

Example H0E50

```
> T2 := HeckeOperator(Jzero(100),2);
> DecomposeUsing(T2);
[
    Modular abelian variety of dimension 1 and level 2^2*5^2 over Q,
    Modular abelian variety of dimension 5 and level 2^2*5^2 over Q,
    Modular abelian variety of dimension 1 and level 2^2*5^2 over Q
]
> W := AtkinLehnerOperator(Jzero(100),100);
> DecomposeUsing(W);
[
    Modular abelian variety of dimension 3 and level 2^2*5^2 over Q,
    Modular abelian variety of dimension 4 and level 2^2*5^2 over Q
```

```
]
```

0.7.4 Additional Examples

Example H0E51

We compute a decomposition of $J_0(46)$ as a product of simple abelian subvarieties.

```
> J := Jzero(46); J;
Modular abelian variety Jzero(46) of dimension 5 and level 2*23 over Q
> Decomposition(J);
[
    Modular abelian variety 46A of dimension 1, level 2*23 and
    conductor 2*23 over Q,
    Modular abelian variety N(23,46,1)(23A) of dimension 2, level
    2*23 and conductor 23^2 over Q,
    Modular abelian variety N(23,46,2)(23A) of dimension 2, level
    2*23 and conductor 23^2 over Q
]
```

Thus J decomposes as a product $E \times A \times B$, where E is an elliptic curve of conductor 46, and A and B are two isogenous images of $J_0(23)$.

```
> J(1);
Modular abelian variety 46A of dimension 1, level 2*23 and
conductor 2*23 over Q
> Conductor(J(1));
46
> Factorization(Conductor(J(2)));
[ <23, 2> ]
```

The `Factorization` command gives an explicit decomposition with embeddings of each factor into $J_0(46)$.

```
> Factorization(Conductor(J(2)));
[ <23, 2> ]
> Factorization(J);
[*
<Modular abelian variety 46A of dimension 1, level 2*23 and
conductor 2*23 over Q, [
    Homomorphism from 46A to Jzero(46) given on integral homology
    by:
    [ 1  0 -2 -1 -1  1  1  1 -2  1]
    [ 0  1 -1 -1  0  0  0  1 -1  0]
]>,
<Modular abelian variety 23A of dimension 2, level 23 and
conductor 23^2 over Q, [
    Homomorphism N(23,46,1) from 23A to Jzero(46) given on integral
```

```
    homology by:
    [-1  1 -1  1  0 -1 -1  1 -1  2]
    [ 0  0 -1  2 -2 -1  0  0  1  0]
    [ 0  0  0  1 -2  0  0  1  0  0]
    [ 0  1  0 -1  0  0  1  0  0  0],
    Homomorphism N(23,46,2) from 23A to Jzero(46) given on integral
    homology by:
    [ 0 -1  0  0  1 -1  0  1  0  0]
    [-1  0  0  0  0 -1  2 -1  1 -1]
    [-1  1 -1  0  0  0  2 -2  2 -2]
    [ 0  0 -1  2 -1  0  0  0  0 -1]
]>
*]
```

0.8 Orthogonal Complements

0.8.1 Complements

The following two commands find a complement of an abelian subvariety of an abelian variety. Existence of a complement is guaranteed by the Poincare reducibility theorem (if we were just working with n-dimensional complex tori, then there need not be complements of subtori). MAGMA computes a complement using the module-theoretic structure of the ambient variety and, in some cases, the intersection pairing on homology.

`Complement(A : parameters)`

`IntPairing` BOOLELT *Default : $false$*

The complement of the image of A under the first embedding of A (the first map in the sequence `Embeddings(A)`).

`ComplementOfImage(phi : parameters)`

`IntPairing` BOOLELT *Default : $false$*

Suppose $\phi : A \to B$ is a morphism of abelian varieties. By the Poincare reducibility theorem, there is an abelian variety C such that $\phi(A) + C = B$ and the intersection of $\phi(A)$ with C is finite. This intrinsic returns a choice of C and an embedding of C into B.

Example H0E52

We compute a decomposition of $J_0(33)$ as a product of simples, then find a decomposition of the complement of one of the factors.

```
> J := Jzero(33);
> D := Decomposition(J); D;
[
    Modular abelian variety 33A of dimension 1, level 3*11 and
```

```
    conductor 3*11 over Q,
    Modular abelian variety N(11,33,1)(11A) of dimension 1, level
    3*11 and conductor 11 over Q,
    Modular abelian variety N(11,33,3)(11A) of dimension 1, level
    3*11 and conductor 11 over Q
]
> A := D[3];
> B := Complement(A);
> B;
Modular abelian variety of dimension 2 and level 3*11 over Q
> Decomposition(B);
[
    Modular abelian variety image(33A) of dimension 1, level 3*11
    and conductor 3*11 over Q,
    Modular abelian variety image(11A) of dimension 1, level 3*11
    and conductor 11 over Q
]
```

Here we compute a somewhat random map from $J_0(11)$ to $J_0(33)$, and compute the complement of the image.

```
> phi := 2*NaturalMap(Jzero(11),Jzero(33),1) - 3*NaturalMap(Jzero(11),Jzero(33),3);
> phi;
Homomorphism from Jzero(11) to Jzero(33) given on integral homology by:
[ 2  6 -7 -2 -6  3]
[ 5 -2 -3 -1 -1  5]
> C,pi := ComplementOfImage(phi);
> C;
Modular abelian variety of dimension 2 and level 3*11 over Q
> Decomposition(C);
[
    Modular abelian variety image(33A) of dimension 1, level 3*11
    and conductor 3*11 over Q,
    Modular abelian variety image(11A) of dimension 1, level 3*11
    and conductor 11 over Q
]
```

0.8.2 Dual abelian variety

This command computes the abelian variety dual to A. It doesn't currently work in full generality, but should work in many cases of interest. Suppose the modular map $A \to J$ is injective, where J is attached to a space of modular symbols and J is isomorphic to its dual (e.g., $J = J_0(N)$). To compute the dual of A, we find a complement B of A in J whose homology is orthogonal to the homology of A with respect to the intersection pairing. This can frequently be accomplished (e.g., when A is attached to a newform) without using the intersection pairing by find a complement B such that the rational homology of B as a

module over the Hecke algebra has no simple factors in common with that of A. Then J/B is isomorphic to the dual of A.

`CanComputeDual(A)`

True if know how to compute the dual of A, and the dual. Otherwise, false and an error message.

`Dual(A)`

The dual abelian variety of A. Currently we require that the modular map to a modular symbols abelian variety is injective.

`ModularPolarization(A)`

The polarization on A induced by pullback of the theta divisor.

Example H0E53

We compute the dual of a 2-dimensional newform abelian variety of level 43, and note that it is isomorphic to itself.

```
> J := Jzero(43);
> A := Decomposition(J)[2]; A;
Modular abelian variety 43B of dimension 2, level 43 and
conductor 43^2 over Q
> Adual := Dual(A); Adual;
Modular abelian variety of dimension 2 and level 43 over Q
> IsIsomorphic(A,Adual);
true Homomorphism from 43B to modular abelian variety of
dimension 2 given on integral homology by:
[-1  1 -1  1]
[-1  0  1  0]
[-1  0  0  0]
[ 0  0 -1  1]
```

Next we compute the dual of a 2-dimensional newform abelian variety of level 69, and find that it is not isomorphic to itself.

```
> A := Decomposition(Jzero(69))[2]; A;
Modular abelian variety 69B of dimension 2, level 3*23 and
conductor 3^2*23^2 over Q
> Adual := Dual(A); Adual;
Modular abelian variety of dimension 2 and level 3*23 over Q
> IsIsomorphic(A,Adual);
false
```

One can show that the natural map from A to its dual is a polarization of degree 484.

```
> phi := NaturalMap(A,Adual);
> phi;
Homomorphism N(1) from 69B to modular abelian variety of
dimension 2 given on integral homology by:
```

```
[  3   1   5  -7]
[ -1   5  -1   1]
[ -6   4  -1   7]
[ 11  -7   6 -12]
> Degree(phi);
484
> factor(484);
[ <2, 2>, <11, 2> ]
1
```

0.8.3 Intersection pairing

These commands compute the matrix of the intersection pairing on homology with respect to the fixed basis for rational or integral homology. If A is not a modular symbols abelian variety (such as $J_0(N)$, then the intersection pairing on homology computed below *is the one got by pulling back from the one on the codomain of the modular embedding* of A. This may not be what you expect, but is easy to compute in great generality.

Computation of intersection pairings is currently only implemented for weight 2.

`IntersectionPairing(A)`

The intersection pairing matrix on the basis for the rational homology of H, pulled back using the modular embedding.

`IntersectionPairing(H)`

The intersection pairing matrix on the basis for the rational homology of H.

`IntersectionPairingIntegral(A)`

The intersection pairing matrix on the basis for the integral homology of H, pulled back using the modular embedding.

Example H0E54

The intersection pairing on $J_0(11)$ is very simple.

```
> J := Jzero(11);
> IntersectionPairing(J);
[ 0 -1]
[ 1  0]
> IntersectionPairingIntegral(J);
[ 0 -1]
[ 1  0]
```

The intersection pairing associated to $J_0(33)$ is more interested. Note that the representing matrix is skew symmetric and has determinant 1.

```
> I := IntersectionPairingIntegral(Jzero(33)); I;
```

```
[ 0  1  0  1  0  1]
[-1  0  0  1  0  1]
[ 0  0  0  1  0  1]
[-1 -1 -1  0  0  1]
[ 0  0  0  0  0  1]
[-1 -1 -1 -1 -1  0]
> Determinant(I);
1
```

The intersection pairing on **33A** is surprising, because it is pulled back from the intersection pairing on $J_0(33)$. Thus instead of having determinant 1, it has determinant 9. [[TODO: I could probably figure out how to change this so the true intersection pairing is computed, but I'm not immediately sure how!? This would be desirable, since there is an intersection pairing on the A below that has determinant 1.]]

```
> A := ModularAbelianVariety("33A");  A;
Modular abelian variety 33A of dimension 1 and level 3*11 over Q
> I := IntersectionPairingIntegral(A); I;
[ 0  3]
[-3  0]
> Determinant(I);
9
```

0.8.4 Projections

Suppose ϕ is a homomorphism from A to B. Then the image $\phi(A)$ is an abelian subvariety of B. The commands below compute a map π in the endomorphism algebra of B whose image is $\phi(A)$ and such that $\pi^2 = \pi$, i.e., π is projection onto $\phi(A)$. A projection map is not canonical, but if the optional parameter `int_pairing` is set to true, then projection is also required to respect the intersection pairing, which uniquely determines π.

`ProjectionOnto(A :` *parameters*`)`

`IntPairing` BoolElt *Default :* $false$

`ProjectionOntoImage(phi :` *parameters*`)`

`IntPairing` BoolElt *Default :* $false$

Let $\phi : A \to B$ be a morphism. This intrinsic computes a projection onto $\phi(A)$ as an element of the endomorphism ring tensor **Q**. If the optional parameter IntPairing is set, then this is the canonical orthogonal projection.

Example H0E55

```
> pi := ProjectionOnto(ModularAbelianVariety("33A"));  pi;
Homomorphism pi from Jzero(33) to Jzero(33) (up to isogeny) on
integral homology by:
 (not printing 6x6 matrix)
> Matrix(pi);
[ 2/3  1/3 -1/3  1/3    0 -2/3]
[ 1/3  2/3 -2/3  2/3    0 -1/3]
[ 1/3  1/3 -1/3  1/3    0 -1/3]
[   0  2/3 -2/3  2/3    0    0]
[   0  1/3 -1/3  1/3    0    0]
[-1/3  1/3 -1/3  1/3    0  1/3]
> pi^2 eq pi;
true
> Rank(pi);
1

> phi := NaturalMap(Jzero(11),Jzero(44));
> pi := ProjectionOntoImage(phi); pi;
Homomorphism pi from Jzero(44) to Jzero(44) (up to isogeny) on
integral homology by:
 (not printing 8x8 matrix)
> A := Image(5*pi); A;
Modular abelian variety of dimension 1 and level 2^2*11 over Q
> IsIsomorphic(Jzero(11),A);
true Homomorphism from Jzero(11) to modular abelian variety of
dimension 1 given on integral homology by:
[ 0 -1]
[-1  1]
```

0.8.5 Left and right inverses

The `LeftInverse` and `RightInverse` commands compute left and right inverses in the category of abelian varieties up to isogeny. The corresponding commands with `Morphism` appended compute a left or right inverse times a minimal integer, so that the result is a homomorphism.

MAGMA computes a right inverse of a finite-degree homomorphism ϕ by finding the projection map onto the complement of the image of ϕ, and composing with a inverse from the image. To find a left inverse of a surjective homomorphism $\phi : A \to B$, MAGMA computes the complement C of the kernel of ϕ; this complement C is an abelian subvariety of A that maps isomorphically onto B, and MAGMA finds the left inverse by inverting ϕ restricted to C.

`LeftInverse(phi : parameters)`

`IntPairing` BoolElt *Default : false*

A homomorphism $\psi : B \to A$ of minimal degree in the category of abelian varieties up to isogeny such that $\psi * \phi$ is the identity map on B. Here $\phi : A \to B$ is a surjective homomorphism.

`LeftInverseMorphism(phi : parameters)`

`IntPairing` BoolElt *Default : false*

A homomorphism $\psi : B \to A$ of minimal degree such that $\psi * \phi$ is multiplication by an integer, where $\phi : A \to B$ is a surjective homomorphism.

`RightInverse(phi : parameters)`

`IntPairing` BoolElt *Default : false*

A map $\psi : B \to A$ in the category of abelian varieties up to isogeny such that $\phi * \psi : A \to A$ is the identity map. Here $\phi : A \to B$ is a homomorphism of abelian varieties with finite kernel.

`RightInverseMorphism(phi : parameters)`

`IntPairing` BoolElt *Default : false*

A minimal-degree homomorphism $\psi : B \to A$ such that $\phi * \psi : A \to A$ is multiplication by an integer, where $\phi : A \to B$ is a homomorphism of abelian varieties with finite kernel.

Example H0E56

First we compute the difference ϕ of the two natural degeneracy maps $J_0(11) \to J_0(33)$, which has as kernel a group of order 5 (called the Shimura subgroup in this case).

```
> Jone1 := Jzero(11);  J33 := Jzero(33);
> d1 := NaturalMap(Jone1,J33,1);
> d3 := NaturalMap(Jone1,J33,3);
> phi := d1-d3;
> Degree(phi);
5
```

A right inverse of ϕ is a homomorphism up to isogeny from $J_0(33)$ to $J_0(11)$.

```
> RightInverse(phi);
Homomorphism from Jzero(33) to Jzero(11) (up to isogeny) on integral
homology by:
 (not printing 6x2 matrix)
15
> RightInverseMorphism(phi);
Homomorphism from Jzero(33) to Jzero(11) (not printing 6x2 matrix)
> phi*RightInverseMorphism(phi);
Homomorphism from Jzero(11) to Jzero(11) given on integral homology by:
```

```
[15  0]
[ 0 15]
```

Finally we find a left inverse of a map from $J_0(33)$ to $J_0(11)$.

```
> psi := NaturalMap(J33,Jone1,1) - NaturalMap(J33,Jone1,3);
> IsSurjective(psi);
true
> LeftInverse(psi);
Homomorphism from Jzero(11) to Jzero(33) (up to isogeny) on integral
homology by:
[ 1/5    0 -2/5  2/5 -3/5    0]
[ 1/5 -1/5    0  1/5 -2/5  1/5]
5
> LeftInverseMorphism(psi);
Homomorphism from Jzero(11) to Jzero(33) given on integral homology by:
[ 1  0 -2  2 -3  0]
[ 1 -1  0  1 -2  1]
> LeftInverseMorphism(psi)*psi;
Homomorphism from Jzero(11) to Jzero(11) given on integral homology by:
[5 0]
[0 5]
```

0.8.6 Congruence computations

The two commands below each give an integer that measures "congruences" between an abelian variety and other abelian varieties. These two quantities are related because if a prime divides the modular degree, then it divides the congruence modulus, though the converse need not be true (see the example below).

If $A = A_f$ is an abelian variety attached to a newform f, then the `CongruenceModulus` command computes the congruence modulus of the newform f, which is an integer that measures congruences between f and nonconjugate forms in the Peterson complement of f. More precisely, if $f \in S_k(N, \varepsilon)$, which is the direct sum of the spaces of modulus forms with character a Galois conjugate of ε, then we define the congruence modulus of f to be the order of the group $S_k(N, \varepsilon; \mathbf{Z})/(W + W^{\perp})$, where W is the intersection of $S_k(N, \varepsilon; \mathbf{Z})$ with the span of the Galois conjugates of f.

Suppose $A \subset J_0(N)$. Then the modular degree of A is the square root of the degree of the induced polarization $A \to J_0(N) \to A'$. Similar remarks apply with $J_0(N)$ replaced by other abelian varieties. Also, when the weight is bigger than 2, we do not take a square root.

`CongruenceModulus(A)`

If A is attached to a newform, this returns the congruence modulus of the newform, taken in the space $S_2(N, \epsilon)$, where ϵ is the character of the newform..

`ModularDegree(A)`

The modular degree of A. This is the square root of the degree of the degree of the map from the dual A' to A. In some cases where no algorithm is implemented for computing A', a message is printed and the square of the degree of the composition of the modular embedding with the modular parameterization is computed. When a weight is bigger than 2 the square root is not taken.

Example H0E57

The modular degree and congruence modulus of one of the two elliptic curves of conductor 54 are interesting because they are not equal. This is the smallest level of an elliptic curve where these two invariants differ. (For more details, see [Agashe-Stein] – TODO.)

```
> J := Jzero(54);
> A,B := Explode(Decomposition(NewSubvariety(J)));
> ModularDegree(A);
6
> CongruenceModulus(A);
6
> ModularDegree(B);
2
> CongruenceModulus(B);
6
```

The modular degree and congruence modulus are 4 for a certain abelian surface A of level 65. We also compute the kernel of the modular map and see that it is $A[2]$.

```
> J := Jzero(65);
> A := J(2); A;
Modular abelian variety 65B of dimension 2, level 5*13 and
conductor 5^2*13^2 over Q
> CongruenceModulus(A);
4
> ModularDegree(A);
4
> phi := NaturalMap(A,Dual(A));
> Invariants(Kernel(phi));
[ 2, 2, 2, 2 ]
```

0.9 New and Old Subvarieties and Natural Maps

0.9.1 Natural maps

Suppose M and N are positive integers and M divides N. There are natural maps in both directions between $J_0(N)$ and $J_0(M)$ (and likewise for J_1, etc.), for each divisor of N/M, which correspond to maps of the form $f(q)$ maps to $f(q^t)$ and their duals. Since any modular abelian variety A in MAGMA is equipped with a map $A \to J_e$ and $J_p \to A$, where J_e and J_p are attached to modular symbols, the problem of defining natural maps between A and B is reduced to defining natural maps between modular abelian varieties attached to modular symbols. The command `NaturalMaps` computes a sequence of natural maps from A to B, corresponding to appropriate divisors of relevant levels. The command `NaturalMap` just returns the natural map corresponding to the divisor 1. The command `NaturalMap` can also be given a third argument, which specifies the divisor.

`NaturalMap(A, B)`

The natural map induced by the identity on modular forms, or 0 if there is none.

`NaturalMap(A, B, d)`

The natural map from A to B induced, in a potentially complicated way, from the map $f(q) \mapsto f(q^d)$ on modular forms. In situations where the modular forms associated to A and B have nothing to do with each other, then we define this map to be the zero map.

`NaturalMaps(A, B)`

Sorted list of maps `NaturalMap(A,B,d)`, where d runs through all divisors of the level of A over the level B, or the level of B over the level A.

Example H0E58

```
> A := Jzero(11)*Jzero(22);
> B := Jzero(11)*Jzero(33);
> phi := NaturalMap(A,B);
> phi;
Homomorphism N(1) from Jzero(11) x Jzero(22) to Jzero(11) x Jzero(33) (not
printing 6x8 matrix)
> Nullity(phi);
1
> f := NaturalMap(A,B,3); f;
Homomorphism N(3) from Jzero(11) x Jzero(22) to Jzero(11) x Jzero(33) (not
printing 6x8 matrix)
> Nullity(f);
2
> NaturalMaps(Jzero(11),Jzero(33));
[
    Homomorphism N(1) from Jzero(11) to Jzero(33) given on integral
```

```
    homology by:
    [ 1  0 -2  2 -3  0]
    [ 1 -1  0  1 -2  1],
    Homomorphism N(3) from Jzero(11) to Jzero(33) given on integral
    homology by:
    [ 0 -2  1  2  0 -1]
    [-1  0  1  1 -1 -1]
]
```

If we take a product of several copies of $J_0(11)$ and of several copies of $J_0(22)$, the `NaturalMaps` command still only returns 2 natural maps, one for each divisor of the quotient of the levels.

```
> A := Jzero(11)^2;
> B := Jzero(22)^3;
> NaturalMaps(A,B);
[
    Homomorphism N(1) from Jzero(11) x Jzero(11) to Jzero(22) x Jzero(22) x
    Jzero(22) given on integral homology by:
    [ 0  1 -2  3  0  1 -2  3  0  1 -2  3]
    [ 1 -1  1  0  1 -1  1  0  1 -1  1  0]
    [ 0  1 -2  3  0  1 -2  3  0  1 -2  3]
    [ 1 -1  1  0  1 -1  1  0  1 -1  1  0],
    Homomorphism N(2) from Jzero(11) x Jzero(11) to Jzero(22) x Jzero(22) x
    Jzero(22) given on integral homology by:
    [-1  0  2 -2 -1  0  2 -2 -1  0  2 -2]
    [-1  2 -1  0 -1  2 -1  0 -1  2 -1  0]
    [-1  0  2 -2 -1  0  2 -2 -1  0  2 -2]
    [-1  2 -1  0 -1  2 -1  0 -1  2 -1  0]
]
```

0.9.2 New subvarieties and quotients

These commands compute the new and r-new subvarieties and quotients of an abelian variety A of level N. The *r-new subvariety* of A is the intersection of the kernels of all natural maps from A to modular abelian varieties of level the level N/r. The *new subvariety* is the intersection of the r-new subvarieties over all prime divisors r of N. The *r-new quotient* of A is the quotient of A by the sum of all images in A under all natural maps of abelian varieties of level N/r.

`NewQuotient(A)`

The new quotient of A.

`NewQuotient(A, r)`

The r-new quotient of A.

`NewSubvariety(A)`

The new subvariety of A.

`NewSubvariety(A, r)`

The r-new subvariety of A.

Example H0E59

```
> J := Jzero(33);
> Dimension(J);
3
> Dimension(NewSubvariety(J,3));
1
> Dimension(NewSubvariety(J));
1
> Dimension(NewSubvariety(J,11));
3
> Dimension(NewQuotient(J));
1
> Dimension(OldSubvariety(J));
2
> Dimension(OldSubvariety(J,3));
2
```

0.9.3 Old subvarieties and quotients

These commands compute the old and r-old subvarieties and quotients of an abelian variety A of level N. The *r-old subvariety* of A is the sum of the images of all natural maps from modular abelian varieties of level N/r to A. The *old subvariety* is the sum of the r-old subvarieties as r varies over the divisors of N. The r-old quotient of A is the quotient of A by its r-new subvariety.

`OldQuotient(A)`

The old quotient of A.

`OldQuotient(A, r)`

The r-old quotient of A.

`OldSubvariety(A)`

The old subvariety of A.

`OldSubvariety(A, r)`

The r-old subvariety of A.

Example H0E60

We compute the old subvariety and old quotient of $J_0(100)$, both of which have dimension 6.

```
> J := Jzero(100); J;
Modular abelian variety Jzero(100) of dimension 7 and level 2^2*5^2
over Q
> J_old := OldSubvariety(J); J_old;
Modular abelian variety Jzero(100)_old of dimension 6 and level
2^2*5^2 over Q
> phi := Embeddings(J_old)[1];
> Codomain(phi);
Modular abelian variety Jzero(100) of dimension 7 and level 2^2*5^2
over Q
> Jold := OldQuotient(J); Jold;
Modular abelian variety Jzero(100)^old of dimension 6 and level
2^2*5^2 over Q
```

The new subvariety and new quotient of $J_0(100)$ intersect in a finite subgroup isomorphic to $\mathbf{Z}/12\mathbf{Z} \times \mathbf{Z}/12\mathbf{Z}$.

```
> J_new := NewSubvariety(J); J_new;
Modular abelian variety Jzero(100)_new of dimension 1 and level
2^2*5^2 over Q
> G, A := J_new meet J_old; G;
Finitely generated subgroup of abelian variety with invariants
[ 12, 12 ]
> Dimension(A);
0
```

0.10 Elements of Modular Abelian Varieties

We represent torsion points on modular abelian varieties as follows. Suppose A is an abelian variety defined over the complex numbers $\mathbf{C}$. Then $A(\mathbf{C})$ is canonically isomorphic to $H_1(A, \mathbf{R})/H_1(A, \mathbf{Z})$, and the torsion subgroup of $A(\mathbf{C})$ is isomorphic to $H_1(A, \mathbf{Q})/H_1(A, \mathbf{Z})$. We represent a torsion element of $A(\mathbf{C})$ by giving a representative element of $H_1(A, \mathbf{Q})$. The functions below provide basic arithmetic operations with such elements, application of homomorphisms, and conversion functions.

Sometimes it is useful to consider elements of $H_1(A, \mathbf{R})$, given by floating point vectors (i.e., over `RealField()`). These represent certain points of infinite order, but without further information we do not know exactly what point they represent, or even whether such a point is 0.

0.10.1 Arithmetic

The following commands support basic arithmetic operations on elements of modular abelian varieties. Operations include addition, subtraction, and multiplication by an integer, rational number, or real number.

`a * x`

Product of the real number a by the element x.

`a * x`

Product of the rational number a by the element x.

`a * x`

Product of the integer a by the element x.

`x * a`

Product of the real number a by the element x.

`x * a`

Product of the rational number a by the element x.

`x * a`

Product of the integer a by the element x.

`x + y`

The sum of x and y.

`x - y`

The difference x minus y.

Example H0E61

In this example, we construct $J_0(23)$, and consider the finite subgroup $\ker(T_3 - 5)$, which has order 400. We then do various arithmetic operations with some of its elements.

```
> A := Jzero(23);
> t3 := HeckeOperator(A,3);
> Factorization(CharacteristicPolynomial(t3));
[
    <x^2 - 5, 2>
]
> G := Kernel(t3-5);
> #G;
400
> Generators(G);
[
    Element of abelian variety defined by [1/10 0 1/10 1/5] modulo homology,
    Element of abelian variety defined by [0 0 0 -5/2] modulo homology,
    Element of abelian variety defined by [1/10 -1/10 0 -1/5] modulo homology,
    Element of abelian variety defined by [1 -3/2 2 1] modulo homology
]
```

```
> x := G.1;
> 1.5*x;
Element of abelian variety defined by [0.14999999999999999999999999998 0.E-28
0.14999999999999999999999999998
0.29999999999999999999999999996] modulo homology
> (3/2)*x;
Element of abelian variety defined by [3/20 0 3/20 3/10] modulo homology
> 10*x;
0
> x*1.5;
Element of abelian variety defined by [0.14999999999999999999999999998 0.E-28
0.14999999999999999999999999998
0.29999999999999999999999999996] modulo homology
> 1.5*x eq x*1.5;
true
> x*(3/2);
Element of abelian variety defined by [3/20 0 3/20 3/10] modulo homology
> x*5;
Element of abelian variety defined by [1/2 0 1/2 1] modulo homology
> G.1 + G.2;
Element of abelian variety defined by [1/10 0 1/10 -23/10] modulo homology
> G.1 - G.2;
Element of abelian variety defined by [1/10 0 1/10 27/10] modulo homology
```

0.10.2 Invariants

These commands compute information about the order of an element, the degree of the homology of the parent variety, and a field that the point is defined over.

`ApproximateOrder(x)`

The exact order of x, if x is known exactly as a torsion point, and if not the order of an approximation of x by a torsion point, obtained using continued fractions.

`Degree(x)`

The dimension of the homology of the parent of x.

`FieldOfDefinition(x)`

A field that x is defined over, which need not be minimal.

`Order(x)`

The order of x, if x is known exactly. Otherwise an error occurs.

Example H0E62

We compute a 2-torsion point on the elliptic curve $J_0(11)$, compute some approximate orders, and compute the degree.

```
> A := Jzero(11);
> G := Kernel(nIsogeny(A,2));
> G;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2 ]
> x := G.1;
> ApproximateOrder(Sqrt(2)*x);
175568277047523
> ApproximateOrder(1.000000000000001*x);
2
> Degree(x);
2
```

Notice that `FieldOfDefinition(x)` is valid, but far from optimal. It would be better to return the number field generated by the 2-torsion point.

```
> FieldOfDefinition(x);
Algebraically closed field with no variables
> FieldOfDefinition(0*x);
Rational Field
> Order(x);
2
```

0.10.3 Predicates

These are commands for testing equality, inclusion, whether an element is 0, and whether an element is known exactly, (i.e., as an element of $H_1(A, \mathbf{Q})$, or just as an element of $H_1(A, \mathbf{R})$).

`IsExact(x)`

True if x is known exactly, i.e., x is defined by an element of the rational homology.

`IsZero(x)`

True if x is known exactly and is equal to 0. If x is not known exactly, true if a real homology vector that represents x is "very close" to an element of the integral homology, where very close means that the distance is within $1/10^n$, where `n=M‘point_precision` and M is the parent of x.

`x eq y`

True if x equals y.

`x in X`

True if x in an element of the list X.

Example H0E63

We demonstrate each of these commands using elements of the 2-torsion subgroups of the two elliptic curves of conductor 37.

```
> J := Jzero(37);
> A, B := Explode(Decomposition(J));
> A;
Modular abelian variety 37A of dimension 1, level 37 and
conductor 37 over Q
> B;
Modular abelian variety 37B of dimension 1, level 37 and
conductor 37 over Q
> A2 := Kernel(nIsogeny(A,2));
> B2 := Kernel(nIsogeny(B,2));
> x := A2.1;
> y := B2.2;
> x eq y;
false
> x in [* x, y *];
true
> IsZero(x);
false
> IsZero(0*x);
true
> IsExact(1.00000000000000000001*x);
false
> IsExact((2/3)*x);
true
```

For non-exact elemenets, `IsZero` means "is quite close to 0".

```
> IsZero(0.0001*x);
false
> IsZero(0.00001*x);
true
> IsZero(0.000000000001*x);
true
```

0.10.4 Homomorphisms

There are two notations for applying a homomorphism to an element. Also, one can find an inverse image of an element using the @@ command.

`phi(x)`

The image of x under the homomorphism ϕ.

`x * phi`

The image of x under the homomorphism ϕ.

`x @@ phi`

An inverse image of x under the homomorphism ϕ.

Example H0E64

Let $\phi = T_3 - 5$ acting on the abelian surface $J_0(23)$. We apply ϕ to an element of the kernel G of ϕ, and get 0. We also find an element y such that $\phi(y)$ is a certain element of G.

```
> A := Jzero(23);
> phi := HeckeOperator(A,3) - 5;
> G := Kernel(phi);
> x := G.1;
> Order(x);
10
> phi(x);
0
> zero := A!0;
> z := zero@@phi; z;
0
> y := x@@phi; y;
Element of abelian variety defined by [-1/20 1/20 -1/20 -1/20] modulo homology
> phi(y) in G;
true
> y*phi eq phi(y);
true
```

0.10.5 Representation of Torsion Points

`ApproximateByTorsionPoint` finds an exact torsion point representation for an element of a modular abelian variety; it uses continued fractions to find good rational approximations for each coordinate of a representative real homology class. The `Element` command gives a representive element of the homology, and the `LatticeCoordinates` command returns a representative of homology written with respect to a basis for integral homology. The `Eltseq` command gives the sequence of entries of the vector returned by `Element`.

`ApproximateByTorsionPoint(x : parameters)`

`Cutoff` RngIntElt *Default :* 10^3

If x is defined by an element z in the real homology $H_1(A, R)$, use continued fractions to find an element of $H_1(A, \mathbf{Q})$ that approximates z, and return corresponding point.

`Element(x)`

The vector in homology that represents x.

`Eltseq(x)`

A sequence of rational or real numbers that defines x.

`LatticeCoordinates(x)`

A vector over the rational or real field that represents x with respect to the homology of the parent abelian variety of x.

Example H0E65

This code illustrates each of the commands for a 3-torsion point in $J_0(33)$.

```
> A := Jzero(33);
> x := A![1/3,0,0,0,0,0];
> x;
Element of abelian variety defined by [1/3 0 0 0 0 0] modulo homology
> Order(x);
3
> ApproximateByTorsionPoint(1.001*x);
Element of abelian variety defined by [1001/3000 0 0 0 0 0] modulo homology
> Element(x);
(1/3   0   0   0   0   0)
> Eltseq(x);
[ 1/3, 0, 0, 0, 0, 0 ]
> LatticeCoordinates(x);
(1/3   0   0   0   0   0)
```

The `Element` and `LatticeCoordinates` can differ when the integral structure on the homology is complicated. This is common when the weight is bigger than 2.

```
> A := Jzero(11,4); A;
Modular motive Jzero(11,4) of dimension 2 and level 11 over Q
> x := A![1/3,0,0,0];
> Element(x);
(  1/8  1/24 -1/24 -1/24)
> Eltseq(x);
[ 1/3, 0, 0, 0 ]
> LatticeCoordinates(x);
(1/3   0   0   0)
> x;
Element of abelian variety defined by [1/3 0 0 0] modulo homology
```

0.11 Subgroups of Modular Abelian Varieties

0.11.1 Creation

The `Subgroup` command constructs the subgroup generated by an arbitrary sequence of elements of an abelian variety A. The `nTorsionSubgroup` command create the n-torsion subgroup $A[n]$ of an abelian variety or of a subgroup. Other common ways to create subgroups are as kernels of homomorphisms, and by taking an image of the difference of two cusps (see the `CuspidalSubgroup` and `RationalCuspidalSubgroup` commands).

If a subgroup G contains elements that are not known exactly (i.e., they are defined by floating point approximations to real homology elements), then `ApproximateByTorsionGroup` can be used to find a group of torsion points that approximates G.

`ApproximateByTorsionGroup(G : parameters)`

`Cutoff` RngIntElt *Default :* 10^3

The subgroup generated by torsion approximations of a set of generators of G.

`Subgroup(X)`

The subgroup of A generated by the nonempty sequence X of elements of a modular abelian variety.

`ZeroSubgroup(A)`

The zero subgroup of the abelian variety A.

`nTorsionSubgroup(A, n)`

The kernel $A[n]$ of the multiplication by n isogeny on A.

`nTorsionSubgroup(G, n)`

The kernel $G[n]$ of the multiplication by n homomorphism on G.

Example H0E66

First we list the elements of the 2-torsion subgroup of the elliptic curve 100A, then we compute the 0 subgroup.

```
> A := ModularAbelianVariety("100A"); A;
Modular abelian variety 100A of dimension 1 and level 2^2*5^2
over Q
> G := nTorsionSubgroup(A,2); G;
Finitely generated subgroup of abelian variety with invariants [ 2, 2 ]
> Elements(G);
[
    0,
```

```
    Element of abelian variety defined by [1/2 0] modulo homology,
    Element of abelian variety defined by [0 1/2] modulo homology,
    Element of abelian variety defined by [1/2 1/2] modulo homology
]
> ZeroSubgroup(A);
{ 0 }:  finitely generated subgroup of abelian variety with
invariants []
```

We can also use the `nTorsionSubgroup` command on subgroups.

```
> nTorsionSubgroup(G,2);
Finitely generated subgroup of abelian variety with
invariants [ 2, 2 ]
> nTorsionSubgroup(G,3);
{ 0 }:  finitely generated subgroup of abelian variety with
invariants []
```

One of the 2-torsion elements generates a subgroup H of order 2.

```
> G.1;
Element of abelian variety defined by [0 1/2] modulo homology
> H := Subgroup([G.1]); H;
Finitely generated subgroup of abelian variety
> #H;
2
```

To illustrate the approximation command, we consider the subgroup generated by an approximation to one of the 2-torsion elements.

```
> K := Subgroup([1.00001*G.1]);
> L := ApproximateByTorsionGroup(K);
Finitely generated subgroup of abelian variety with
invariants [ 2 ]
> L eq H;
true
```

0.11.2 Elements

These commands enumerate elements of a finite subgroup of a modular abelian variety, and also allow standard access to the elements.

`Elements(G)`

Sequence of all elements of the finite subgroup G of a modular abelian variety.

`Generators(G)`

Sequence of generators of G. These correspond to generators for the underlying abelian group.

`Ngens(G)`

The number generators of G.

`G . i`

The i-th generator of G.

Example H0E67

We illustrate each of the commands using the kernel of the Hecke operator T_3 acting on $J_0(67)$.

```
> J := Jzero(67);
> T := HeckeOperator(J,3);
> G := Kernel(T);
> #G;
4
> Elements(G);
[
    0,
    Element of abelian variety defined by [0 0 1/2 0 0 -1/2 -1/2 0 1/2 0] modulo homology,
    Element of abelian variety defined by [1/2 -1/2 0 0 -1/2 0 0 -1/2 1/2 0] modulo
homology,
    Element of abelian variety defined by [1/2 -1/2 1/2 0 -1/2 -1/2 -1/2 -1/2 1 0] modulo
homology
]
> Generators(G);
[
    Element of abelian variety defined by [1/2 -1/2 0 0 -1/2 0 0 -1/2 1/2 0] modulo
homology,
    Element of abelian variety defined by [0 0 1/2 0 0 -1/2 -1/2 0 1/2 0] modulo homology
]
> Ngens(G);
2
> G.1;
Element of abelian variety defined by [1/2 -1/2 0 0 -1/2 0 0 -1/2 1/2 0] modulo homology
> G.2;
Element of abelian variety defined by [0 0 1/2 0 0 -1/2 -1/2 0 1/2 0] modulo homology
```

0.11.3 Arithmetic

These commands support taking quotients of abelian varieties by finite subgroups, intersecting finite subgroups with other finite subgroups or abelian varieties, and computing the group generated by two subgroups.

For several of the arithmetic operations below, finite groups or abelian varieties are replaced by their image in a common abelian variety, so the operation makes sense. This common abelian variety is the one returned by the `FindCommonEmbeddings` command. Note that the "embedding" is only guaranteed to be an embedding up to isogeny.

`Quotient(A, G)`

Quotient of the abelian variety A by the finite subgroup G, the isogeny $A \to A/G$ and an isogeny $A/G \to A$, such that composition of the two isogenies is multiplication by the exponent of G.

`Quotient(G)`

The quotient A/G, where A is the ambient variety of G, an isogeny from A to A/G with kernel G, and an isogeny from A/G to A such that the composition of the two isogenies is multiplication by the exponent of G.

`A / G`

The quotient A/G, the isogeny $A \to A/G$ with kernel G, and an isogeny $A/G \to A$.

`A meet G`

The intersection of the finite subgroup G of an abelian variety B with the abelian variety A. If A is not equal to B, then G and A are replaced by their image in a common abelian variety.

`G meet A`

The intersection of the finite subgroup G of an abelian variety B with the abelian variety A. If A is not equal to B, then G and A are replaced by their image in a common abelian variety.

`G1 + G2`

The sum of the subgroups groups G_1 and G_2 of abelian varieties A_1 and A_2. If A_1 is not equal to A_2, then G_1 and G_2 are replaced by their image in a common abelian variety.

`G1 meet G2`

The intersection of the finite subgroups G_1 and G_2 of an abelian variety. If their ambient varieties are not equal, G_1 and G_2 are replaced by their image in a common abelian variety.

Example H0E68

We illustrate these commands using the 2-torsion of $J_0(67)$. First we compute the kernel of T_3, which is a 2-torsion group of order 4.

```
> J := Jzero(67); J;
Modular abelian variety Jzero(67) of dimension 5 and level 67 over Q
> T := HeckeOperator(J,3);
> Factorization(CharacteristicPolynomial(T));
[
    <x + 2, 2>,
    <x^2 - x - 1, 2>,
    <x^2 + 3*x + 1, 2>
]
> G := Kernel(T); #G;
```

```
4
```

Next we quotient $J_0(67)$ out by this subgroup of order 4.

```
> A := Quotient(J,G); A;
Modular abelian variety of dimension 5 and level 67 over Q
```

The result is, of course, isogenous to $J_0(67)$. (TODO: Unfortunately, testing of isomorphism in this generality is not yet implemented.)

```
> IsIsogenous(A,J);
true
> Degree(ModularParameterization(A));
4
```

If the `Quotient` command is given only one argument then the variety being quotiented out by is the ambient variety.

```
> B := Quotient(G); B;
Modular abelian variety of dimension 5 and level 67 over Q
> Degree(ModularParameterization(B));
4
```

We can also use the divides notation for quotients.

```
> C := J/G; C;
Modular abelian variety of dimension 5 and level 67 over Q
```

Next we list the 2-torsion subgroups of the simple factors of $J_0(67)$. Interestingly, the sum of the 2-torsion subgroups of these simple factors is much smaller than the full 2-torsion subgroup $J_0(67)[2]$.

```
> D := Decomposition(J); D;
[
    Modular abelian variety 67A of dimension 1, level 67 and
    conductor 67 over Q,
    Modular abelian variety 67B of dimension 2, level 67 and
    conductor 67^2 over Q,
    Modular abelian variety 67C of dimension 2, level 67 and
    conductor 67^2 over Q
]
> for A in D do print #(A meet G); end for;
4
1
1
> G2 := nTorsionSubgroup(D[2],2);
> G3 := nTorsionSubgroup(D[3],2);
> H := G + G2 + G3;
> #H;
64
> H eq nTorsionSubgroup(J,2);
false
```

```
> #nTorsionSubgroup(J,2);
1024
> G2 eq G3;
true
> G meet G2;
{ 0 }:  finitely generated subgroup of abelian variety with invariants []
```

0.11.4 Underyling abelian group and lattice

Let G be a finitely generated subgroup of an abelian variety A. The `AbelianGroup` command returns an abstract abelian group that is isomorphic to G along with isomorphisms in both directions.

Assume that G is a torsion group, with all elements known exactly. The `Lattice` command returns a lattice L in the rational homology of A such that L/H equal G, where we identify the torsion in A as the rational homology modulo homology.

`AbelianGroup(G)`

An abstract abelian group H isomorphic to G, a map from H to G, and a map from G to H.

`Lattice(G)`

Assume G is a finite torsion subgroup of its ambient abelian variety A and G is generated by elements of $H_1(A, \mathbf{Q})/H_1(A, \mathbf{Z})$. Viewing G as a set of equivalence classes of the form $x + H_1(A, \mathbf{Z})$, this command returns the lattice generated by $H_1(A, \mathbf{Z})$ and all such x.

Example H0E69

This examples illustrate these commands for the 3-torsion subgroup of $J_0(11)$.

```
> A := Jzero(11);
> G := nTorsionSubgroup(A,3);
> H,f,g := AbelianGroup(G);
> H;
Abelian Group isomorphic to Z/3 + Z/3
Defined on 2 generators
Relations:
    3*H.1 = 0
    3*H.2 = 0
```

The lattice of G is 1/3 times the integral homology.

```
> Lattice(G);
Lattice of rank 2 and degree 2
Basis:
[Identity matrix]
```

```
Basis Denominator:  3
> L := IntegralHomology(A); L;
Standard Lattice of rank 2 and degree 2
> Lattice(G)/L;
Abelian Group isomorphic to Z/3 + Z/3
Defined on 2 generators
Relations:
    3*$.1 = 0
    3*$.2 = 0
```

0.11.5 Invariants

Let G be a finitely generated subgroup of an abelian variety. The `AmbientVariety` command gives the abelian variety whose elements were used to create G.

The `Exponent` command returns the smallest positive integer e such that $eG = 0$. The `Invariants` command returns the invariants of an abstract abelian group isomorphic to G. The `Order` command (or $\#G$) returns the number of elements in G, when G is known to be finite (or an error otherwise).

The `FieldOfDefinition` command gives a field over which the group G is defined. This is a field K so that if σ is an automorphism that fixes K, then $\sigma(G) = G$. Note that K *is not guaranteed to be minimal*.

`AmbientVariety(G)`

Abelian variety that contains G.

`Exponent(G)`

An integer that kills G. We assume G is finite.

`FieldOfDefinition(G)`

A field over which the group G is defined (this is not guaranteed to be minimal!).

`Invariants(G)`

Invariants of G as a finite abelian group.

`Order(G)`

The number of elements in G.

`#G`

The number of elements in G.

Example H0E70

We illustrate each command using the kernel of T_3 on $J_0(67)$.

```
> A := Jzero(67);
> T3 := HeckeOperator(A,3);
> G := Kernel(T3); G;
Finitely generated subgroup of abelian variety with
[ 2, 2 ]
> AmbientVariety(G);
Modular abelian variety Jzero(67) of dimension 5 and level 67 over Q
> Exponent(G);
2
> Invariants(G);
[ 2, 2 ]
> Order(G);
4
> #G;
4
```

The field of definition of G is $\mathbf{Q}$, since G is the kernel of a homomorphism defined over $\mathbf{Q}$ (a Hecke operator).

```
> FieldOfDefinition(G);
Rational Field
```

However, the field of definition of the subgroup of G generated by one of the elements of G could take significant extra work to determine. Currently MAGMA simply chooses the easiest answer, which is $\overline{\mathbf{Q}}$.

```
> H := Subgroup([G.1]);
> FieldOfDefinition(H);
Algebraically closed field with no variables
```

0.11.6 Predicates and comparisons

The `IsFinite` command is true exactly when every element of G is known exactly, since then all elements are torsion and G is finitely generated.

The `subset` commands check inclusion and `eq` checks equality. Equality and subset testing is liberal, in that if the ambient varieties containing the two groups are not equal, then MAGMA attempts to find a natural embedding of both subgroups into a common ambient variety, and checks equality or inclusion there.

`IsFinite(G)`

True if G is known to be finite, e.g., generated by torsion elements. If G is not known exactly, i.e., has elements defined by floating point approximations to homology, then this is false.

`A subset G`

True if the abelian variety A is a subset of the finitely generated group G. Thus this is true only if A is a point, i.e., the 0 dimensional abelian variety.

`G subset A`

True if G is a subset of the abelian variety A. If A is not the ambient variety of G, then G and A are first mapped to a common ambient variety and compared.

`G1 eq G2`

True if G_1 equals G_2.

`G1 subset G2`

True if $G1$ is a subset of $G2$.

Example H0E71

We work with $J_0(389)$, but work in the $+1$ quotient of homology for efficiency. First we let A and B be the first and fifth factors in the decomposition of J, and let G and H be the corresponding 5-torsion subgroups.

```
> J := Jzero(389,2,+1);
> D := Decomposition(J);
> A := D[1];
> B := D[5];
> G := nTorsionSubgroup(A,5);
> H := nTorsionSubgroup(B,5);
```

Note that the torsion subgroups aren't as big because we are working in the $+1$ quotient.

```
> #G;
5
> #H;
95367431640625
```

We now demonstrate each of the above commands for A, B, G, and H.

```
> IsFinite(G);
true
> A subset G;
false
> ZeroModularAbelianVariety() subset G;
true
> G subset A;
true
> G subset B;
true
```

```
> H subset A;
false
```

Since the ambient varieties of G and H are A and B, respectively, the following commands implicitly embed G and H into $J_0(389)$ and make comparisons there.

```
> G subset H;
true
> G eq H;
false
> G eq G;
true
```

```
> J := Jzero(37);
> A, B := Explode(Decomposition(J));
> A2 := Kernel(nIsogeny(A,2));
> B2 := Kernel(nIsogeny(B,2));
> A2 eq B2;
true
> x := A2.1;
> x in B2;   // uses embedding of both into $J_0(37)$.
true
```

0.12 Rational Torsion Subgroups

The following functions are used for computing information about certain torsion points on modular abelian varieties.

0.12.1 Cuspidal subgroup

For simplicity, assume A is a modular abelian variety and $\pi : J_0(N) \to A$ is the modular parameterization (the case when $J_0(N)$ is replaced by a more general modular abelian variety is similar). The *cuspidal subgroup* of $J_0(N)$ is the finite torsion group generated by all classes of differences of cusps on $X_0(N)$. The *cuspidal subgroup* of $A(\overline{\mathbf{Q}})$ is the image under π of the cuspidal subgroup of $J_0(N)$. The *rational cuspidal subgroup* is the subgroup generated by differences of cusps that are defined over $\mathbf{Q}$. (Computation of the rational points in the cuspidal subgroup has not yet been implemented.) One important use of the rational cuspidal subgroup is that it gives a lower bound on the cardinality (and structure) of the torsion subgroup of $A(\mathbf{Q})$, which is important in computations involving the Birch and Swinnerton-Dyer conjecture.

`CuspidalSubgroup(A)`

The subgroup of A generated by all differences of cusps, where we view A as a quotient of a modular symbols abelian variety. Note that this subgroup need not be defined over $\mathbf{Q}$.

`RationalCuspidalSubgroup(A)`

Finite subgroup of A generated by all differences of $\mathbf{Q}$-rational cusps, where we view A in some way as a quotient of a modular symbols abelian variety.

Example H0E72

We compute the cuspidal and rational cuspidal subgroups of $J_0(100)$.

```
> J := Jzero(100);
> G := CuspidalSubgroup(J); G;
Finitely generated subgroup of abelian variety with invariants
[ 6, 30, 30, 30, 30 ]
> [Eltseq(x) :  x in Generators(G)];
[
    [ 29/30, -2/5, 16/15, 121/30, -2, -61/30, 3/5, 31/15, 1,
    -89/30, -7/2, -3/2, -3, -1 ],
    [ 1, -5/6, 0, -1, -1, 2/3, -3/2, 0, -2, 0, 2, 5/3, 5/6, 0 ],
    [ -2, 17/15, 1/10, -29/15, 89/30, 26/15, 7/10, -2, 2/5, 2,
    -3/10, -7/30, 59/30, -1/2 ],
    [ 29/30, -1, 1/2, 67/15, -2, -3/2, 14/15, 91/30, 1, -3,
    -38/15, -29/30, -91/30, 1/2 ],
    [ 31/30, -31/30, 2/5, 67/15, -29/15, -43/30, 5/6, 3, 1, -3,
    -13/5, -31/30, -91/30, 0 ]
]
> H := RationalCuspidalSubgroup(J); H;
Finitely generated subgroup of abelian variety with invariants
[ 3, 15, 30 ]
```

Next we compute the cuspidal and rational subgroups for the optimal new elliptic curve of conductor 100.

```
> D := Decomposition(J); A := D[1];
> CuspidalSubgroup(A);
Finitely generated subgroup of abelian variety with invariants
[ 2, 2 ]
> Generators(CuspidalSubgroup(A));
[
    Element of abelian variety defined by [0 1/2] modulo homology,
    Element of abelian variety defined by [1/2 0] modulo homology
]
> RationalCuspidalSubgroup(A);
Finitely generated subgroup of abelian variety with invariants []
> TorsionMultiple(A);
```

```
2
```

Because the torsion multiple is 2, some of the cuspidal subgroup can not be defined over $\mathbf{Q}$.

0.12.2 Upper and lower bounds

Let A be an abelian variety over a number field K. The `TorsionLowerBound` command computes a divisor of the cardinality of the torsion subgroup of $A(K)$. Currently, to compute a bound we require that A be the base extension of an abelian variety B over $\mathbf{Q}$, and the lower bound is simply the cardinality of the rational cuspidal subgroup of $B(\mathbf{Q})$.

The `TorsionMultiple` command computes a multiple of the cardinality of the torsion subgroup of $A(K)$. This multiple is usually fairly sharp, and is computed as follows. For each good prime p with $[K : Q] + 1 < p$, MAGMA computes $\#A(k)$, where k varies over residue class fields of K of characteristic p. Since reduction on torsion is injective for such primes, the greatest common divisor of the $\#A(k)$ is a multiple of the order of the torsion subgrop of $A(K)$. MAGMA computes $\#A(k)$ by using Hecke operators to find the characteristic polynomial of Frobenius on a Tate module of A, and uses this characteristic polynomial to deduce $\#A(k)$. (TODO: details, see "that paper with Amod".)

`TorsionLowerBound(A)`

A divisor of the cardinality of the K-rational torsion subgroup of A over K.

`TorsionMultiple(A)`

Same as $TorsionMultiple(A, 50)$.

`TorsionMultiple(A, n)`

A multiple of the cardinality of the K-rational torsion subgroup of A over K obtained by counting points on A mod p, where p varies over the odd primes $p \leq n$ such that p does not divide the level of A.

Example H0E73

```
> J := Jzero(100);
> TorsionLowerBound(J);
1350
> #RationalCuspidalSubgroup(J);
1350
> TorsionMultiple(J);
16200
> 16200/1350;
12
> J2 := BaseExtend(J,QuadraticField(2));
> TorsionMultiple(J2);
129600
```

```
> 129600/16200;
8
```

0.12.3 Torsion Subgroup

Let A be an abelian variety over a field K. The `TorsionSubgroup` command attempts to compute the subgroup of torsion elements in $A(K)$.

`TorsionSubgroup(A)`

Either false and a subgroup of the torsion subgroup, or true and the exact torsion subgroup of A over the base field.

Example H0E74

```
> TorsionSubgroup(Jzero(11));
true Finitely generated subgroup of abelian variety with invariants [ 5 ]
> TorsionSubgroup(Jzero(33));
false Finitely generated subgroup of abelian variety with
invariants [ 10, 10 ]
> TorsionSubgroup(BaseExtend(Jzero(11),QuadraticField(5)));
true Finitely generated subgroup of abelian variety with
invariants [ 5 ]
> TorsionSubgroup(ChangeRing(Jzero(11),GF(5)));
false { 0 }:  finitely generated subgroup of abelian variety with
invariants []
> TorsionSubgroup(Jzero(100));
false Finitely generated subgroup of abelian variety with
invariants [ 3, 15, 30 ]
> TorsionSubgroup(Jzero(125));
true Finitely generated subgroup of abelian variety with
invariants [ 25 ]
```

0.13 Hecke and Atkin-Lehner Operators

0.13.1 Creation

These commands compute endomorphisms induced by the Atkin-Lehner and Hecke operators on modular abelian varieties. The Atkin-Lehner involution W_q is defined for each positive integer q that exactly divides the level (and is divisible by the conductor of any relevant character).

`AtkinLehnerOperator(A)`

The morphism (or morphism tensor Q) on (or from) A induced by the Atkin-Lehner operator.

`AtkinLehnerOperator(A, q)`

The Atkin-Lehner operator W_q of index n induced on A by virtue of A being modular. In general W_q need not be a morphism except in the category of abelian varieties up to isogeny so this intrinsic also returns an integer d such that $d * W_q$ is an endomorphism, and when W_q doesn't leave A invariant, also returns $d = 0$. If the ambient modular symbols space of A contains a space with character of conductor r, then currently an error occurs unless r divides q.

`HeckeOperator(A, n)`

The Hecke operator T_n of index n induced on A by virtue of its morphism to a modular symbols abelian variety. In general T_n need not be a morphism. Also, if A is contained in e.g., `Jzero(N),then` the T_n on $J_0(N)$ need not even leave A invariant. In that case this command composes T_n with a map back to A to obtain an endomorphism of A. For the exact Hecke operators induced by their action on `Jzero(N)`, say, use the RestrictEndomorphism command.

Example H0E75

We compute the main Atkin-Lehner operator and the Hecke operator T_2 on $J_0(23)$.

```
> A := Jzero(23);
> AtkinLehnerOperator(A,23);
Homomorphism W23 from Jzero(23) to Jzero(23) given on integral homology by:
[-1  0  0  0]
[ 0 -1  0  0]
[ 0  0 -1  0]
[ 0  0  0 -1]
> HeckeOperator(A,2);
Homomorphism T2 from Jzero(23) to Jzero(23) given on integral homology by:
[ 0  1 -1  0]
[ 0  1 -1  1]
[-1  2 -2  1]
[-1  1  0 -1]
```

Next we compute w_4 and w_{25} on J_{100}, and note that their product equals w_{100}.

```
> A := Jzero(100); A;
Modular abelian variety Jzero(100) of dimension 7 and
level 2^2*5^2 over Q
> w4 := AtkinLehnerOperator(A,4);
> Factorization(CharacteristicPolynomial(w4));
[
    <x - 1, 4>,
    <x + 1, 10>
]
> w25 := AtkinLehnerOperator(A,25);
> Factorization(CharacteristicPolynomial(w25));
[
    <x - 1, 8>,
    <x + 1, 6>
]
> w4*w25 eq AtkinLehnerOperator(A);
true
```

Next we compute W_{25} acting on $J_1(25)$.

```
> A := Js(17);
> B := BaseExtend(A,CyclotomicField(17));
> w := AtkinLehnerOperator(B);
> Factorization(CharacteristicPolynomial(w));
[
    <x - 1, 4>,
    <x + 1, 6>
]
```

Finally we compute Hecke operators on the quotient of a simple factor of $J_0(65)$ by a finite subgroup.

```
> A := Decomposition(Jzero(65))[2]; A;
Modular abelian variety 65B of dimension 2, level 5*13 and conductor
5^2*13^2 over Q
> G := nTorsionSubgroup(A,2); G;
Finitely generated subgroup of abelian variety with invariants
[ 2, 2, 2, 2 ]
> H := Subgroup([G.1]); H;
Finitely generated subgroup of abelian variety with invariants [ 2 ]
> B := A/H; B;
Modular abelian variety of dimension 2 and level 5*13 over Qbar
> T2 := HeckeOperator(B,2); T2;
Homomorphism from modular abelian variety of dimension 2 to
modular abelian variety of dimension 2 (up to isogeny) on
integral homology by:
[ -2 1/2   0   0]
[ -2   2   0   0]
```

```
[ -2   1  -2   1]
[ -6   1  -1   2]
> FactoredCharacteristicPolynomial(T2);
[
    <x^2 - 3, 2>
]
```

0.13.2 Invariants

The `HeckePolynomial` and `FactoredHeckePolynomial` commands compute characteristic polynomials and factored characteristic polynomials of Hecke operators. The `MinimalHeckePolynomial` command computes minimal polynomials of Hecke operators.

`FactoredHeckePolynomial(A, n)`

The factored characteristic polynomial of the Hecke operator T_n acting on A. This can be faster than first computing T_n, then computing the characteristic polynomial, and factoring, because we can take into account information about the decomposition of A, in order to avoid factoring.

`HeckePolynomial(A, n)`

The characteristic polynomial of the Hecke operator T_n acting on A.

`MinimalHeckePolynomial(A, n)`

The minimal polynomial of the Hecke operator T_n acting on A.

Example H0E76

```
> FactoredHeckePolynomial(Jzero(65),2);
[
    <x + 1, 2>,
    <x^2 - 3, 2>,
    <x^2 + 2*x - 1, 2>
]
> HeckePolynomial(Jzero(65),2);
x^10 + 6*x^9 + 5*x^8 - 32*x^7 - 62*x^6 + 28*x^5 + 130*x^4 +
    48*x^3 - 51*x^2 - 18*x + 9
> MinimalHeckePolynomial(Jzero(65),2);
x^5 + 3*x^4 - 2*x^3 - 10*x^2 - 3*x + 3
```

0.14 L-series

0.14.1 Creation

The `LSeries` command creates the L-series $L(A, s)$ associated to a modular abelian variety A over $\mathbf{Q}$ or a cyclotomic field. No actual computation is performed.

`LSeries(A)`

The L-series associated to A.

Example H0E77

```
> A := Jzero(23);
> L := LSeries(A);
> L;
L(Jzero(23),s):  L-series of Modular abelian variety Jzero(23) of
dimension 2 and level 23 over Q
> LSeries(ModularAbelianVariety("65B"));
L(65B,s):  L-series of Modular abelian variety 65B of dimension 2
and level 5*13 over Q
```

You can create L-series of abelian varieties over cyclotomic fields, but currently no interesting functionality is implemented for them. [TODO: something for abelian extensions using twists and characters.]

```
> LSeries(BaseExtend(Jzero(11),CyclotomicField(5)));
L(Jzero(11),s):  L-series of Modular abelian variety Jzero(11) of
dimension 1 and level 11 over Q(zeta_5)
```

0.14.2 Invariants

`CriticalStrip(L)`

Integers x and y so that the critical strip for L is the set of complex numbers with real part strictly between x and y. If W is the set of weights of newforms that give rise to factors of A, where $L = LSeries(A)$, then this command returns 0 and `Max(W)`.

`ModularAbelianVariety(L)`

The abelian variety to which L was attached.

Example H0E78

We define several L-functions of modular abelian varieties and modular motives, and compute their critical strip (which is from 0 to k, where k is the weight).

```
> L := LSeries(Jzero(37));
> CriticalStrip(L);
0 2
> L := LSeries(Jzero(37,6));
> CriticalStrip(L);
0 6
> J := Jone(11,3);  J;
Modular motive Jone(11,3) of dimension 5 and level 11 over Q
> CriticalStrip(LSeries(J));
0 3
> A_delta := Jzero(1,12);
> L := LSeries(A_delta);
> CriticalStrip(L);
0 12
> ModularAbelianVariety(L);
Modular motive Jzero(1,12) of dimension 1 and level 1 over Q
```

0.14.3 Characteristic polynomials of Frobenius elements

Let A be a modular abelian variety. The characteristic polynomials of Frobenius elements acting on the ℓ-adic Tate modules of A define the local L-factors of $L(A, s)$.

`FrobeniusPolynomial(A : parameters)`

`factored` BoolElt *Default : false*

The characteristic polynomial of Frobenius on A.

`FrobeniusPolynomial(A, P)`

The characteristic polynomial of Frobenius at the nonzero prime ideal P on the modular abelian variety A, where p is assumed to be a prime of good reduction for A, and A is defined over a field that contains the prime P.

`FrobeniusPolynomial(A, p : parameters)`

`factored` BoolElt *Default : false*

The characteristic polynomial of Frob_p acting on any ell-adic Tate module of A, where p and ell do not divide the level of A. Here A is an abelian variety over a number field. If the base ring has degree bigger than 1, then return a sequence of charpolys, one for each prime lying over p, sorted by degree.

Example H0E79

```
> A := Jzero(23);
> FrobeniusPolynomial(A,2);
x^4 + x^3 + 3*x^2 + 2*x + 4
> A := Jzero(23) * Jzero(11,4) * Jone(13);
> FrobeniusPolynomial(A,2);
x^12 + 2*x^11 + 17*x^10 + 40*x^9 + 145*x^8 + 362*x^7 + 798*x^6 +
    1408*x^5 + 2104*x^4 + 2528*x^3 + 2528*x^2 + 1792*x + 1024
> Factorization($1);
[
    <x^4 - 2*x^3 + 14*x^2 - 16*x + 64, 1>,
    <x^4 + x^3 + 3*x^2 + 2*x + 4, 1>,
    <x^4 + 3*x^3 + 5*x^2 + 6*x + 4, 1>
]
> A := BaseExtend(Jzero(23),CyclotomicField(22));
> FrobeniusPolynomial(A,2);
[
    x^4 + 25*x^3 - 327*x^2 + 25600*x + 1048576
]
```

These characteristic polynomials are used in the algorithm to compute the number of points on modular abelian varieties over finite fields.

```
> A := ChangeRing(Jzero(23),GF(2^10));
> NumberOfRationalPoints(A);
1073875 1073875
> factor($1);
[ <5, 3>, <11, 2>, <71, 1> ]
```

0.14.4 Values at integers in the critical strip

The following commands compute the special value $L(A, s)$, where s is an integer that lies strictly within the critical strip. Also, `LRatio` computes a well-defined quotient $L(A, s)/\alpha$ that lies in the rational numbers. The `LRatio` command requires that A be an abelian variety over $\mathbf{Q}$ attached to a newform.

There exist algorithms for computing $L(A, s)$ for any complex number s, but these are not currently implemented in MAGMA. [TODO? – at least give references.] [[TODO: say something or add code for computing L-functions of elliptic curves and higher derivatives.]]

`L(s)`

The value of L at s, where s is an integer that lies in the critical strip.

`Evaluate(L, s)`

The value of L at s, where s must be an integer that lies in the critical strip for L.

`Evaluate(L, s, prec)`

The value of L at s, where s must be an integer that lies in the critical strip for L, computed using prec terms of power series.

`IsZeroAt(L, s)`

`LRatio(A, s)`

`LRatio(L, s)`

The ratio $L(A, j) * (j-1)!/((2\pi)^{j-1} * \Omega_j)$, where j is a "critical integer", so $1 \leq j \leq k-1$, and Ω_j is the volume of the group of real points on A when j is odd, and the volume of the -1 eigenspace for conjugation when j is even.

Example H0E80

First we demonstrate each evaluation command for the L-series of $J_0(23)$.

```
> L := LSeries(Jzero(23));
> L(1);
0.248431866590599284683305769290 +  0.E-29*i
> Evaluate(L,1);
0.248431866590599284683305769290 +  0.E-29*i
> Evaluate(L,1,200);
0.248431866590599681207250339074 +  0.E-29*i
> LRatio(L,1);
1/11
> L := LSeries(Jzero(23));
> L(1);
0.248431866590599284683305770476 +  0.E-29*i
> Evaluate(L,1,200);
0.248431866590599681207250340144 +  0.E-29*i
```

Next we compute the L-series of the motive attached to the weight 12 level 1 modular form Δ.

```
> A := Jzero(1,12);
> L := LSeries(A);
> Evaluate(L,1);
0.037441281268515541738770315844 3
> L(5);
0.666709188434003643826130221 64
> Evaluate(L,1,200);
0.037441281268515541738770315844 3
> LRatio(L,1);
11340/691
> LRatio(L,2);
24
> LRatio(L,3);
7
```

We compute some ratios for $J_1(N)$ and factors of $J_1(N)$.

```
> LRatio(Jone(13),1);
1/361
> J := Jone(23);
> Evaluate(LSeries(J),1);
0.00000008077769707478577542009070006 6 +
0.00000005367962127748221777320766933 2*i
```

It looks kind of like $L(J_1(23), 1)$ is zero. However, this is not the case! We can not compute `LRatio` for $J_1(23)$, since it not attached to a newform. We can, however, compute `LRatio` for each simple factor.

```
> LRatio(J(1),1);
1/11
> LRatio(J(2),1);
1/1382426761
```

Each simple factor has nonzero `LRatio`, so $L(J, 1) \neq 0$.

0.14.5 Leading coefficient

The L-function $L(A, s)$ has a Taylor expansion about any critical integer. The `LeadingCoefficient` command computes the leading coefficient and order of vanishing of $L(A, s)$ about this critical integer. [TODO: say more about how, when it works, how it's restricted to low rank.]

`LeadingCoefficient(L, s, prec)`

The leading coefficient of the Taylor expansion about the critical integer s and the order of vanishing of L at s. At present, the abelian variety that defines L must have weight 2 and trivial character (so $s = 1$). It does not have to be attached to a newform.

Example H0E81

```
> LeadingCoefficient(LSeries(Jzero(37)),1,100);
0.24426406492583898134986778296 5 1
> LeadingCoefficient(LSeries(Jzero(37)(1)) ,1,100);
0.30599977380008529004409407572 5 1
> J := Jzero(3^5);
> LeadingCoefficient(LSeries(J),1,100);
15.140660788463628991688955015326 +  0.E-27*i 4
```

The order of vanishing of 4 for $J_0(3^5)$ comes from an elliptic curve and a 3-dimensional abelian variety that have order of vanishing 1 and 3, respectively.

```
> LeadingCoefficient(LSeries(J(1)),1,100);
1.41920964933821561600318808428 1 1
```

```
> LeadingCoefficient(LSeries(J(5)),1,100);
1.228051952859142052034769858445 3
```

We give a few more examples.

```
> L := LSeries(ModularAbelianVariety("389A",+1));
> LeadingCoefficient(L,1,100);
0.75931650029224679065762600319 2
>
> A := Jzero(65)(2); A;
Modular abelian variety 65B of dimension 2, level 5*13 and
conductor 5^2*13^2 over Q
> L := LSeries(A);
> LeadingCoefficient(L,1,100);
0.91225158869818984109351402175 +  0.E-29*i 0
> A := Jzero(65)(3); A;
Modular abelian variety 65C of dimension 2, level 5*13 and
conductor 5^2*13^2 over Q
> L := LSeries(A);
> LeadingCoefficient(L,1,100);
0.45206792176803106991748613500 0 +  0.E-29*i 0
```

0.15 Complex Period Lattice

0.15.1 Period Map

Let A be a modular abelian variety. The period mapping of A is a map from the rational homology of A to a complex vector space.

`PeriodMapping(A, prec)`

The complex period mapping from rational homology to C^d, where $d = \dim A$, computed using prec terms of q-expansions.

0.15.2 Period Lattice

`Periods(A, n)`

Generators for the complex period lattice of A, computed using n terms of q-expansions. We use the map from A to a modular symbols abelian variety to define the period mapping (so this map must be injective).

0.16 Tamagawa Numbers and Component Groups of Neron Models

0.16.1 Component groups

Suppose A is a newform modular abelian variety over $\mathbf{Q}$ over level N. For any prime p that exactly divides N, the command below computes the order of the component group of B over the algebraic closure of `GF(p)`. The nontrivial algorithm is described in [REFERENCE: Conrad-Stein and Kohel-Stein]. It is an open problem to compute the structure of the component group or the order under more general hypothesis.

`ComponentGroupOrder(A, p)`

The order of the component group of the special fiber of the Neron model of A over the algebraic closure of `GF(p)`. The abelian variety A must be attached to a newform.

Example H0E82

```
> J := Jzero(65); J;
Modular abelian variety Jzero(65) of dimension 5 and level 5*13 over Q
> A := Decomposition(J)[3];
> ComponentGroupOrder(A,13);
1
> ComponentGroupOrder(A,5);
7
```

0.16.2 Tamagawa numbers

Suppose A is an abelian variety over $\mathbf{Q}$ that is attached to a newform. [TODO: Extend to not over Q using Lenstra-Oort bound.] The `TamagawaNumber` command computes a divisor and an integer some power of which is a multiple of the Tamagawa number of A at a prime p. If the optional second argument is omitted then the product of the Tamagawa numbers is computed. [TODO: use lenstra oort, etc.].

`TamagawaNumber(A)`

Let c be the product of the Tamagawa numbers of A at primes of bad reduction, where A is an abelian variety over $\mathbf{Q}$ attached to a newform. This command returns a divisor of c, an integer some power of which is a multiple of c, and true if the divisor is provably equal to c.

`TamagawaNumber(A, p)`

A divisor of the Tamagawa number of A at p, an integer some power of which is a multiple of the Tamagawa number of A at p, and true if the divisor of the Tamagawa number is provably equal to the Tamagawa number of A. The abelian variety A must be attached to a newform.

Example H0E83

```
> J := Jzero(65);
> TamagawaNumber(J(2),5);
2 2 false
> TamagawaNumber(J(2),13);
3 3 true
> TamagawaNumber(J(3),5);
7 7 true
> TamagawaNumber(J(3),13);
2 2 false
> J := Jzero(5^2*7);
> TamagawaNumber(J(1));
2 30 false
> TamagawaNumber(J(1),5);
1 30 false
> TamagawaNumber(J(1),7);
2 2 false
```

0.17 Elliptic curves

0.17.1 Creation

Modular abelian varieties of dimension 1 are elliptic curves. Given a modular abelian variety A over $\mathbf{Q}$ of dimension 1, the first command below computes an elliptic curve that is isogenous over $\mathbf{Q}$ to A. Given an elliptic curve E over $\mathbf{Q}$, the second command returns a modular abelian variety over $\mathbf{Q}$ that is isogenous to E.

It would be very desirable to make these commands more precise, and to extend them to work over other fields. For example, modular abelian varieties should (conjecturally) be associated to $\mathbf{Q}$-curves and their restriction of scalars.

`EllipticCurve(A)`

An elliptic curve isogenous to the modular abelian variety A over the rational field, if there is an elliptic curve associated to A. For A of higher weight use the EllipticInvariants command.

`ModularAbelianVariety(E)`

`ModularAbelianVariety(E, sign)`

A modular abelian variety isogenous to E. Note that elliptic curves with small coefficients can have quite large conductor, hence computing the massive modular abelian variety that has E as quotient, which is one thing this function does, could take some time.

Example H0E84

We apply the above two commands to the elliptic curve $J_0(49)$.

```
> A := Jzero(49);
> E := EllipticCurve(A); E;
Elliptic Curve defined by y^2 + x*y = x^3 - x^2 - 2*x - 1 over
Rational Field
> B := ModularAbelianVariety(E); B;
Modular abelian variety 'Cremona 49A' of dimension 1 and level
7^2 over Q
```

To see how A and B compare, we first test equality and see they are not equal (since they were constructed differently). However, they are isomorphic.

```
> B eq A;
false
> IsIsomorphic(A,B);
true Homomorphism from Jzero(49) to 'Cremona 49A' given on integral
homology by:
[1 0]
[0 1]
> phi := NaturalMap(A,B);
> Degree(phi);
1
> phi;
Homomorphism N(1) from Jzero(49) to 'Cremona 49A' given on integral
homology by:
[1 0]
[0 1]
```

Thus B is embedded in A via the identity map.

0.17.2 Invariants

Let A be an abelian variety over $\mathbf{Q}$ of dimension 1. The following two functions use standard iterative algorithms (see Cremona's book) to compute the invariants c_4, c_6, j, and generators of the period lattice of the optimal quotient of $J_0(N)$ associated to A.

`EllipticInvariants(A, n)`

Invariants c_4, c_6, j, and an elliptic curve, of the one dimensional modular abelian variety A, computed using n terms of q-expansion.

`EllipticPeriods(A, n)`

Elliptic periods w_1 and w_2 of the $J_0(N)$ – *optimal* elliptic curve associated to A, computed using n terms of the q-expansion. The periods have the property that w_1/w_2 has positive imaginary part.

Example H0E85

```
> A := ModularAbelianVariety("100A");
> c4,c6,j,E := EllipticInvariants(A,100);
> c4;
1600.052318304045803306867849111720 8 +  0.E-25*i
> c6;
-44002.166592330033618811790218678607 +  0.E-24*i
> j;
3276.801127299202275905948170653 93 +  0.E-25*i
> E;
Elliptic Curve defined by y^2 = x^3 +
(-43201.412594209236689285431925551172 +  0.E-24*i)*x +
(2376116.995985821815415836671800373 00 +  0.E-22*i) over Complex
Field
> jInvariant(E);
3276.801127299202275905948170705 63 +  0.E-25*i
> w1,w2 := EllipticPeriods(A,100);
> w1;
1.263088700712760693712816573302450091088 +  0.E-38*i
> w2;
 0.E-38 - 1.017029270669959849197879061650056208633 21*i
> w1/w2;
 0.E-38 + 1.241939378874229622446687406094865084049 7*i
```