Manual of the MAGMA package `Weight1`

Gabor Wiese *

8th August 2008

Abstract

This is a short manual for the MAGMA [1] package `Weight1`. The package is the implementation of an algorithm of Edixhoven's for computing weight one modular forms over finite field (see [2], [3], [4], [5]).

The package needs the author's MAGMA package `ArtinAlgebras`. Both packages can be downloaded from the author's webpage.

The package was originally written in 2002 and 2003 and has only been slightly updated since.

Contents

1 Installation and Example

Note that you need the packages `ArtinAlgebras` and `Weight1`. They can be downloaded from the author's webpage. For installation, just unpack the tar-files.

The following example is supposed to quickly explain the main functionalities.

Suppose that `PATH1` contains `ArtinAlgebras.spec` and that `PATH2` contains `Weight1.spec`. Then we attach the packages using

```
> AttachSpec("PATH1/ArtinAlgebras.spec");
> AttachSpec("PATH2/Weight1.spec");
```

Let us take level $N = 1429$ in characteristic 2 for the trivial character. We first create a data structure "whole level".

```
> wl := Create(1429,2);
```

*Institut für Experimentelle Mathematik, Universität Duisburg-Essen, Ellernstr. 29, D-45326 Essen, Germany.
`http://maths.pratum.net/`, e-mail: `gabor.wiese@uni-due.de`

Now we can compute the systems of eigenvalues for all Hecke operators T_n with odd indices. This is where the work is done.

```
> SystemsOfEigenvalues(~wl);
Using integral modular symbols.
Computing intregral Hecke operators.
Computing Hecke algebra generators.
Computing systems of eigenvalues.
```

Our interest is in forms of weight 1 for level 1429. We create them as follows.

```
> ef := EigenformsWt1(wl);
```

We find that there are two of them.

```
> #ef;
2
```

Simple properties can be accessed as follows.

```
> Level(ef[1]);
1429
> Weight(ef[1]);
1
> Characteristic(ef[1]);
2
```

These have the obvious meaning.

```
> FieldDegree(ef[1]);
2
```

This is the degree over $\mathbb{F}_2$ of the field generated by all coefficients.

```
> Dimension(ef[1]);
1
```

This is the dimension (over the coefficient field) of the local factor of the Hecke algebra corresponding to the eigenform.
Let's look at the second form.

```
> FieldDegree(ef[2]);
3
> Dimension(ef[2]);
2
```

The group G of the representation of the eigenform can be obtained like this.

```
> Group(ef[1]);
C_5 SL(2,4)
```

The first result is a subgroup of G, the second a group containing G. Sometimes, one gets the actual group, as we see for the second eigenform.

```
> Group(ef[2]);
SL(2,8) SL(2,8)
```

The local factor of the Hecke algebra corresponding to the system of eigenvalues of the eigenform is obtained by the following command.

```
> H := HeckeAlgebra(ef[1]); H;
Matrix Algebra of degree 1 with 53 generators over GF(2^2)
> H := HeckeAlgebra(ef[2]); H;
Matrix Algebra of degree 2 with 53 generators over GF(2^3)
```

Let us recall that the coefficient field of the second eigenform has degree 3 over $\mathbb{F}_2$. MAGMA denotes a generator of $\mathbb{F}_8$ by `$.1`. We can also study the Hecke operators in the local factor directly.

```
> HeckeOperator(ef[2],7);
[$.1^2    0 ]
[ 0    $.1^2]
```

The corresponding coefficient is the eigenvalue of the operator.

```
> Coefficient(ef[2],7);
$.1^2
> HeckeOperator(ef[2],11);
[ 0    $.1^4 ]
[ $.1^5    0 ]
> Coefficient(ef[2],11);
$.1
```

We can also study the weight 2 eigenforms.

```
> e2 := Eigenforms(wl);
> #e2;
7
```

With them we can proceed precisely as before.

2 Mathematical description

Fix a prime p. The package computes Katz modular forms of weight one for $\Gamma_0(N)$ and some Dirichlet character $\epsilon : (\mathbb{Z}/N\mathbb{Z})^\times \to \overline{\mathbb{F}_p}^\times$. In order to obtain proved results, p should not divide N.
For the algorithm (due to Edixhoven) see [2], [3], [4], [5].

3 Functions

3.1 Eigenforms

For the computation of eigenforms one first has to create the necessary data structure, which will then contain all "systems of eigenvalues" of a given prime weight p, level and character. These data structure are called "whole levels".
In a second step the systems of eigenvalues of the "whole level" for the Hecke operators T_n with $p \nmid n$ are computed.
From those the weight 1 and weight p eigenforms can then be derived.
Finally, the eigenforms and the corresponding local factors of the Hecke algebra can be studied further.

Frequently, the functions and procedures use referenced variables, so that many computations need only be done once. The disadvantage of a more difficult usage seems to be well justified by great gains in speed and memory.

3.1.1 "Whole levels"

intrinsic Create (N :: RngIntElt, k :: RngIntElt) -> Rec
This function creates a "whole level" with level N, weight k. We only require k to be prime and $N > 0$, however, not that N and k are coprime. As the characteristic of the eigenforms is equal to k, this is not allowed in Katz' theory of eigenforms and results in that case are not proved.

intrinsic Create (eps :: GrpDrchElt, k : RngIntElt) -> Rec
This function creates a "whole level" with Dirichlet character eps in weight k. The character must either be defined over a finite field or the rationals. The weight is required to be prime. However, the weight is allowed to divide the modulus. As the characteristic of the eigenforms is equal to k, this is not allowed in Katz' theory of eigenforms and results in that case are not proved.

intrinsic Create (eps :: GrpDrchElt) -> Rec
This function creates a "whole level" with Dirichlet character taking values in a finite field. The weight is set

to be the characteristic of the field. It is allowed that the characteristic divides the modulus of the character, but results in that case are not proved.

3.1.2 Systems of eigenvalues

intrinsic SystemsOfEigenvalues (~WL :: Rec : char := 0, UpToConjugation := true, Wt1APriori := false)
This procedure computes the systems of eigenvalues for all Hecke operators T_n with n coprime to the characteristic, which belong to eigenforms of the given "whole level" WL. This is the longest part of the computation of weight one forms.
The option char can be set to 0 or to a prime p that has to equal the chosen weight. In the first case, integral modular symbols are used for the computation of the Hecke algebra. In the second case, modular symbols over a finite field of characteristic p are used. The second option is faster. Eigenforms in weight one are the same (one of the main results of [5]). In some cases (Eisenstein representations), there are tiny differences of the local Hecke algebras due to 'torsion'.
The option UpToConjugation decides whether all elements of a conjugacy class of systems of eigenvalues are taken, or only one.
If the option Wt1APriori is set, some systems of eigenvalues that cannot correspond to weight 1 are discarded directly. Applying EigenformsWt1 gives a result independent of this option. However, Eigenforms does not. Setting the option leads to a slight speed-up.

3.1.3 Eigenforms of weight p

intrinsic Eigenforms (W, SoEV :: Rec) -> SeqEnum
intrinsic Eigenforms (W :: Rec, L :: SeqEnum) -> SeqEnum
intrinsic Eigenforms (W :: Rec) -> SeqEnum
Given a system of eigenvalues SoEV or a list L of systems of eigenvalues belonging to the "whole level" W, these functions compute a list of all corresponding eigenforms in weight equal to the characteristic. The third call refers to all sytems of eigenvalues belonging to W.

3.1.4 Eigenforms of weight 1

intrinsic EigenformWt1 (W :: Rec, SoEV :: Rec) -> SeqEnum
intrinsic EigenformsWt1 (W :: Rec, SoEV :: Rec) -> SeqEnum
Given a system of eigenvalues SoEV belonging to the "whole level" W, these functions compute a list of all corresponding eigenforms in weight 1. So the list is either empty or contains precisely one element.

intrinsic EigenformsWt1 (W :: Rec, L :: SeqEnum) -> SeqEnum
Given a list L of systems of eigenvalues belonging to the "whole level" W, this function computes a list of all corresponding eigenforms in weight 1.

intrinsic EigenformsWt1 (W :: Rec) -> SeqEnum
Given a "whole level" W, this function computes a list of all eigenforms in weight 1 belonging to the associated systems of eigenvalues.

3.2 Simple properties

intrinsic Level (mf :: Rec) -> RngIntElt
intrinsic Characteristic (mf :: Rec) -> RngIntElt
intrinsic Character (mf :: Rec) -> GrpDrchElt
intrinsic FieldDegree (mf :: Rec) -> RngIntElt
intrinsic Dimension (mf :: Rec) -> RngIntElt
intrinsic Weight (mf :: Rec) -> RngIntElt
These functions return the corresponding values for the eigenform mf.

3.3 Hecke operators and coefficients

An eigenform corresponds to a system of eigenvalues, which in turn corresponds to a maximal ideal of the Hecke algebra. The Hecke operators computed by the following algorithms are the images in the localisation of the Hecke algebra at the maximal ideal. In other words, one can say that the Hecke operators act on the common primary space corresponding to the system of eigenvalues.

intrinsic HeckeOperator (R :: Rec, i :: RngIntElt) -> Mtrx
Given either an eigenform R, or a "whole level" R, this function computes the i-th corresponding Hecke operator for a fixed internal basis. If R is an eigenform, then the operator must have been stored earlier.

intrinsic HeckeOperator (WL :: Rec, mf :: Rec, i :: RngIntElt) -> Mtrx
Given an eigenform mf belonging to a "whole level" WL, this unction computes the i-th corresponding Hecke operator for a fixed internal basis.

intrinsic Coefficient (mf :: Rec, i :: RngIntElt) -> Any
Given an eigenform mf, this function computes the i-th coefficient of the standard q-expansion, provided that the necessary Hecke operators were stored earlier.

intrinsic Coefficient (WL :: Rec, mf :: Rec, i :: RngIntElt) -> Any
Given an eigenform mf belonging to a "whole level" WL, this function computes the i-th coefficient of the standard q-expansion.

3.4 Hecke algebras and bounds

intrinsic HeckeAlgebra (mf :: Rec : bound_factor := 1) -> AlgMat
This function returns the matrix algebra generated by the Hecke operators, acting on the local factor corresponding to the system of eigenvalues of the given eigenform `mf`. Results are proved for `bound_factor := 1`.
As bound for the computation the corresponding Hecke bound will be multiplied by `bound_factor`.

intrinsic HeckeBound (N :: RngIntElt, k :: RngIntElt) -> RngIntElt
intrinsic HeckeBound (eps :: GrpDrchElt, k :: RngIntElt) -> RngIntElt
intrinsic HeckeBound (mf :: Rec) -> RngIntElt
Given a level N and a weight k, this function computes the Hecke bound, which is the smallest integer greater equal $\frac{Nk}{12}\prod_{l|N, l \text{ prime}}(1+\frac{1}{l})$. If a Dirichlet character eps is given, N is taken to be the modulus of eps. If an eigenform mf is given, its level and weight are used.

3.5 The group of a Hecke eigenform

intrinsic Group (mf :: Rec : bound_factor := 2) -> MonStgElt, MonStgElt
Given an eigenform in characteristic 2, this function returns two strings GroupL and GroupU. GroupL contains the name of a lower bound for the group generated by the associated Galois representation. Similarly, GroupU gives an upper bound.
All admissible primes up to `bound_factor` times the corresponding HeckeBound are taken into account.

3.6 Database functions

For performance reasons Hecke operators and coefficients can be stored in the data structures. If a stored operator is requested, it is always taken from memory.
The data structures can also be written into files, so that they can be reused later. It is generally not possible after reloading to compute new Hecke operators since the space of modular symbols used will be lost. However, automatically the Hecke operators up to the Hecke bound will be stored. Consequently, after reloading the Hecke algebra will be available and also the group computation.

3.6.1 Storing Hecke operators and coefficients

intrinsic StoreHeckeOperator (˜WL , i)
Stores the prime operators necessary for the computation of the i-th Hecke operator in the record WL.

intrinsic StoreHeckeOperators (˜WL , a, b)
Stores the prime operators necessary for the computation of the Hecke operator in the range [a..b] in the record WL.

intrinsic StoreHeckeOperators(WL :: Rec, ˜mf :: Rec)
Stores those prime Hecke operators of the eigenform mf in the attribute mf'HeckeList, which are stored for the specified "whole level" WL.

intrinsic StoreHeckeOperator(WL :: Rec, ˜mf :: Rec, i)
Stores the prime Hecke operators of the eigenform mf in the attribute mf'HeckeList, which are necessary for the computation of the i-th Hecke operator.

intrinsic StoreHeckeOperators(WL :: Rec, ˜mf :: Rec, a, b)
Stores the prime Hecke operators of the eigenform mf in the attribute mf'HeckeList, which are necessary for the computation of the Hecke operators in the range [a..b].

intrinsic StoreCoefficients(˜mf :: Rec)
Stores those prime coefficients of the eigenform mf in the attribute mf'CoefficientList, for which the corresponding Hecke operators are stored.

3.6.2 Creating database files

intrinsic WriteFile (mf :: Rec, file :: MonStgElt : Overwrite := false, noOperators := true, noCoefficients := false)
Appends the eigenform mf to the given file. The options noOperators resp. noCoefficients have the obvious meaning. Note, however, that if no operators are stored, then the Hecke algebra structure will be lost.
One can reload the data using:

```
load "Filename";
```

They will be appended to a list L, which msut have been initialised before, e.g. by

```
> L := [];
```

The eigenforms in L are unsorted. However, one can run through the list looking for a certain level.

References

[1] W. Bosma, J. J. Cannon, C. Playoust. *The Magma Algebra System I: The User Language.* J. Symbolic Comput. **24** (1997), pp. 235-265

[2] S. J. Edixhoven. *Comparison of integral structures on spaces of modular forms of weight two, and computation of spaces of forms mod* 2 *of weight* 1. Journal of the Inst. of Math. Jussieu (2006) 5(1), 1-34.

[3] G. Wiese. *Computing Hecke algebras of weight* 1 *in MAGMA*. Appendix B of [2].

[4] G. Wiese. *Modular Forms of Weight One over Finite Fields.* PhD thesis, Universiteit Leiden, 2005. Available from the author's webpage.

[5] G. Wiese. *On the faithfulness of parabolic cohomology as a Hecke module over a finite field.* Journal für die reine und angewandte Mathematik 606 (2007), 79-103.