Using Trace Formulas to Construct Modular Form Spaces

Henri Cohen

Kaiserslautern, September 2, 2016

IMB, Université de Bordeaux

Introduction I

The most commonly used method for computing with spaces of modular forms, largely developed by J. Cremona, W. Stein, and others, is through the use of modular symbols and variants.

Many other methods for computing spaces of modular forms:

- theta functions,
- Eta quotients,
- products of two Eisenstein series,
- Brandt matrices,
- trace formulas,
- explicit representations (N. Skoruppa),
- ...

Introduction I

The most commonly used method for computing with spaces of modular forms, largely developed by J. Cremona, W. Stein, and others, is through the use of modular symbols and variants.

Many other methods for computing spaces of modular forms:

- theta functions,
- Eta quotients,
- products of two Eisenstein series,
- Brandt matrices,
- trace formulas,
- explicit representations (N. Skoruppa),
- ...

Will describe in detail methods using trace formulas, now available in `Pari/GP` in the `GIT` branch

origin/kb-mftrace

Several trace formulas are used:

- Classical: Eichler, Hijikata, and C..
- Use of the theory of Jacobi forms, due to N. Skoruppa and D. Zagier.
- Formula for newforms: J. Bober, A. Booker, and M. Lee, additional formulas by C., implementation help from N. Mascot.

Introduction II

Will describe in detail methods using trace formulas, now available in `Pari/GP` in the `GIT` branch

origin/kb-mftrace

Several trace formulas are used:

- Classical: Eichler, Hijikata, and C..
- Use of the theory of Jacobi forms, due to N. Skoruppa and D. Zagier.
- Formula for newforms: J. Bober, A. Booker, and M. Lee, additional formulas by C., implementation help from N. Mascot.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\mathrm{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\mathrm{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\mathrm{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\mathrm{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Main Tasks

Many different necessary subtasks:

1. Computation of the traces of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (the trace formula proper).
2. Computation of the traces of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ (use of formulas from Bober et al.).
3. Computation of a basis of $S_k^{\text{new}}(\Gamma_0(N),\chi)$, then of a basis of $S_k(\Gamma_0(N),\chi)$.
4. Use of the Hecke algebra to split $S_k^{\text{new}}(\Gamma_0(N),\chi)$ and find the eigenforms.
5. Computation of a basis of Eisenstein series (J. Weisinger's thesis), and projection on $\mathbb{Q}(\chi)$.
6. Computation of the values at the cusps of the Eisenstein series, especially in weight 1.
7. Computation of a basis of weight 1 forms $S_1(\Gamma_0(N),\chi)$ using G. Schaeffer's Hecke stability method.
8. Use of a surprising theorem on the trace form to compute the newspace $S_1^{\text{new}}(\Gamma_0(N),\chi)$ and the eigenforms.

Important Auxiliary Tasks

Also important auxiliary tasks:

1. Evaluation of a modular form at a point in $\mathbb{H}$.
2. Computation of values and special values of the corresponding L-function and symmetric square.
3. Computation of Petersson norms and products.
4. Action of Hecke and Atkin–Lehner operators.
5. Computation of the Fourier expansion at cusps different from $i\infty$.
6. Linear decomposition of a modular form on basis, with or without Eisenstein series.

Almost everything is now implemented.

Talk in three parts:

1. Interesting aspects of the theory;
2. Implementation details;
3. Example commands.

Important Auxiliary Tasks

Also important auxiliary tasks:

1. Evaluation of a modular form at a point in $\mathbb{H}$.
2. Computation of values and special values of the corresponding L-function and symmetric square.
3. Computation of Petersson norms and products.
4. Action of Hecke and Atkin–Lehner operators.
5. Computation of the Fourier expansion at cusps different from $i\infty$.
6. Linear decomposition of a modular form on basis, with or without Eisenstein series.

Almost everything is now implemented.

Talk in three parts:

1. Interesting aspects of the theory;
2. Implementation details;
3. Example commands.

Important Auxiliary Tasks

Also important auxiliary tasks:

1. Evaluation of a modular form at a point in $\mathbb{H}$.
2. Computation of values and special values of the corresponding L-function and symmetric square.
3. Computation of Petersson norms and products.
4. Action of Hecke and Atkin–Lehner operators.
5. Computation of the Fourier expansion at cusps different from $i\infty$.
6. Linear decomposition of a modular form on basis, with or without Eisenstein series.

Almost everything is now implemented.

Talk in three parts:

1. Interesting aspects of the theory;
2. Implementation details;
3. Example commands.

No need to give it precisely here: three aspects.

- It involves class numbers of imaginary quadratic orders. Thus, essential to precompute a sufficiently large table beforehand. Done using classical recursions on Hurwitz class numbers.
- It involves the sum of four multiplicative elementary arithmetic functions. Can either use multiplicativity, or precompute their values. For small level (up to 1000, say), precomputation is faster, larger levels multiplicativity.
- We use a cache method to avoid recomputing already computed traces.

Part I: The Classical Eichler–Selberg Trace Formula

No need to give it precisely here: three aspects.

- It involves class numbers of imaginary quadratic orders. Thus, essential to precompute a sufficiently large table beforehand. Done using classical recursions on Hurwitz class numbers.
- It involves the sum of four multiplicative elementary arithmetic functions. Can either use multiplicativity, or precompute their values. For small level (up to 1000, say), precomputation is faster, larger levels multiplicativity.
- We use a cache method to avoid recomputing already computed traces.

No need to give it precisely here: three aspects.

- It involves class numbers of imaginary quadratic orders. Thus, essential to precompute a sufficiently large table beforehand. Done using classical recursions on Hurwitz class numbers.
- It involves the sum of four multiplicative elementary arithmetic functions. Can either use multiplicativity, or precompute their values. For small level (up to 1000, say), precomputation is faster, larger levels multiplicativity.
- We use a cache method to avoid recomputing already computed traces.

The Trace Formula for Newforms

Initially explained by J. Bober, A. Booker, and M. Lee. Can then invert the formula and express the trace of $T(n)$ on $S_k^{\mathrm{new}}(\Gamma_0(N),\chi)$ in terms of traces on the full cuspidal spaces $S_k(\Gamma_0(M),\chi)$.

Notation: $\mathrm{Tr}(N,n)$ ($\mathrm{Tr}^{\mathrm{new}}(N,n)$) for trace of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (resp., $S_k^{\mathrm{new}}(\Gamma_0(N),n)$) ($k$ and χ are fixed).

$$\mathrm{Tr}^{\mathrm{new}}(N,n)=\sum_{\mathfrak{f}\mid M\mid N}\ \sum_{\substack{d\mid\gcd(M/\mathfrak{f},N_1)\\ d^2\mid n}}\chi_{\mathfrak{f}}(d)d^{k-1}\beta_{n/d^2}(N/M)\,\mathrm{Tr}(M/d,n/d^2).$$

$N=N_1N_2$ with $\gcd(N_1,N_2)=1$, N_1 squarefree, N_2 squarefull, $\mathfrak{f}$ conductor, $\chi_{\mathfrak{f}}$ primitive character equivalent to χ, $\beta_m(N)$ elementary arithmetic function defined by

$$\zeta^{-2}(s)\prod_{p\mid m}(1-1/p^s))=\sum_{N\geq 1}\beta_m(N)/N^s\ .$$

The Trace Formula for Newforms

Initially explained by J. Bober, A. Booker, and M. Lee. Can then invert the formula and express the trace of $T(n)$ on $S_k^{\mathrm{new}}(\Gamma_0(N),\chi)$ in terms of traces on the full cuspidal spaces $S_k(\Gamma_0(M),\chi)$.
Notation: $\mathrm{Tr}(N,n)$ ($\mathrm{Tr}^{\mathrm{new}}(N,n)$) for trace of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (resp., $S_k^{\mathrm{new}}(\Gamma_0(N),n)$) ($k$ and χ are fixed).

$$\mathrm{Tr}^{\mathrm{new}}(N,n)=\sum_{\mathfrak{f}\mid M\mid N}\ \sum_{\substack{d\mid\gcd(M/\mathfrak{f},N_1)\\ d^2\mid n}}\chi_{\mathfrak{f}}(d)d^{k-1}\beta_{n/d^2}(N/M)\,\mathrm{Tr}(M/d,n/d^2).$$

$N=N_1N_2$ with $\gcd(N_1,N_2)=1$, N_1 squarefree, N_2 squarefull, $\mathfrak{f}$ conductor, $\chi_{\mathfrak{f}}$ primitive character equivalent to χ, $\beta_m(N)$ elementary arithmetic function defined by

$$\zeta^{-2}(s)\prod_{p\mid m}(1-1/p^s))=\sum_{N\geq 1}\beta_m(N)/N^s\ .$$

The Trace Formula for Newforms

Initially explained by J. Bober, A. Booker, and M. Lee. Can then invert the formula and express the trace of $T(n)$ on $S_k^{\text{new}}(\Gamma_0(N),\chi)$ in terms of traces on the full cuspidal spaces $S_k(\Gamma_0(M),\chi)$.
Notation: $\mathrm{Tr}(N,n)$ ($\mathrm{Tr}^{\text{new}}(N,n)$) for trace of $T(n)$ on $S_k(\Gamma_0(N),\chi)$ (resp., $S_k^{\text{new}}(\Gamma_0(N),n)$) ($k$ and χ are fixed).

$$\mathrm{Tr}^{\text{new}}(N,n)=\sum_{\mathfrak{f}\mid M\mid N}\ \sum_{\substack{d\mid\gcd(M/\mathfrak{f},N_1)\\ d^2\mid n}}\chi_{\mathfrak{f}}(d)d^{k-1}\beta_{n/d^2}(N/M)\,\mathrm{Tr}(M/d,n/d^2).$$

$N=N_1N_2$ with $\gcd(N_1,N_2)=1$, N_1 squarefree, N_2 squarefull, $\mathfrak{f}$ conductor, $\chi_{\mathfrak{f}}$ primitive character equivalent to χ, $\beta_m(N)$ elementary arithmetic function defined by

$$\zeta^{-2}(s)\prod_{p\mid m}(1-1/p^s))=\sum_{N\geq 1}\beta_m(N)/N^s\ .$$

A Surprising Theorem I

Can go further in new/fullspace computations: set $\mathcal{T}(N)=\sum_{n\ge1}\mathrm{Tr}(N,n)q^n$, and similarly $\mathcal{T}^{\mathrm{new}}$. One can prove the following nontrivial formula:

$$\mathcal{T}(N)=\sum_{\mathfrak{f}\mid M\mid N}\sum_{\substack{D\mid N/M\\ D\text{ cubefree}\\ \gcd(f,M)=1}} c(N/M,D)B(D)T_M(d)\mathcal{T}^{\mathrm{new}}(M)\ ,$$

where for any cubefree integer D we write uniquely $D=df^2$ with d and f squarefree and coprime, and

$$c(N/M,D)=\mu(d)\prod_{p\mid d}v_p(N/M)\chi_{\mathfrak{f}}(f)f^{k-1}\prod_{p\mid f}(v_p(N/M)-1)\sigma_0\left((N/M)_{(D)}\right)\ ,$$

where $N_{(D)}$ is the prime-to-D part of N.

A Surprising Theorem II

Note that $\mathcal{T}^{\mathrm{new}}$ is the sum of the normalized eigenforms, so is trivially in $S_k^{\mathrm{new}}(\Gamma_0(N),\chi) \subset S_k(\Gamma_0(N),\chi)$. The above theorem shows that $\mathcal{T}$ itself is also in $S_k(\Gamma_0(N),\chi)$. Not at all clear a priori since $\mathcal{T}$ involves traces of $T(n)$ with $\gcd(N,n)>1$, well known to behave badly.

Although we do not use it, note a little-known theorem of W. Li: one can modify the definition of $T(n)$ for $\gcd(N,n)>1$ so that the full space of cuspforms (as opposed to the newspace) can be completely diagonalized with multiplicity 1, etc... It is possible that this is related to the theorem mentioned above.

A Surprising Theorem II

Note that $\mathcal{T}^{\mathrm{new}}$ is the sum of the normalized eigenforms, so is trivially in $S_k^{\mathrm{new}}(\Gamma_0(N),\chi) \subset S_k(\Gamma_0(N),\chi)$. The above theorem shows that $\mathcal{T}$ itself is also in $S_k(\Gamma_0(N),\chi)$. Not at all clear a priori since $\mathcal{T}$ involves traces of $T(n)$ with $\gcd(N,n)>1$, well known to behave badly.

Although we do not use it, note a little-known theorem of W. Li: one can modify the definition of $T(n)$ for $\gcd(N,n)>1$ so that the full space of cuspforms (as opposed to the newspace) can be completely diagonalized with multiplicity 1, etc... It is possible that this is related to the theorem mentioned above.

Computing Bases I

Thanks to the above theorem, **two** methods to construct $S_k(\Gamma_0(N), \chi)$.

1. Use $\mathcal{T}(M)$ for all M with $\mathfrak{f} \mid M \mid N$, which are modular forms according to theorem, and apply suitable $B(d)$ and $T(j)$. Advantage: use directly the trace formula. Disadvantage: not at all canonical, inelegant, and in practice slower.
2. Use $\mathcal{T}^{\mathrm{new}}(N)$ and suitable $T(j)$ to construct $S^{\mathrm{new}}(\Gamma_0(N), \chi)$, which we need anyway, and then

$$S_k(\Gamma_0(N), \chi) = \bigoplus_{\mathfrak{f}|M|N} \bigoplus_{d|N/M} B(d) S_k^{\mathrm{new}}(\Gamma_0(M), \chi)\ .$$

Advantage: mostly canonical, elegant. Disadvantage: use of a slightly more complicated trace formula. Nonetheless in practice faster, even for computing the full cuspidal space.

Computing Bases I

Thanks to the above theorem, **two** methods to construct $S_k(\Gamma_0(N),\chi)$.

1. Use $\mathcal{T}(M)$ for all M with $\mathfrak{f} \mid M \mid N$, which are modular forms according to theorem, and apply suitable $B(d)$ and $T(j)$. Advantage: use directly the trace formula. Disadvantage: not at all canonical, inelegant, and in practice slower.
2. Use $\mathcal{T}^{\mathrm{new}}(N)$ and suitable $T(j)$ to construct $S^{\mathrm{new}}(\Gamma_0(N),\chi)$, which we need anyway, and then

$$S_k(\Gamma_0(N),\chi) = \bigoplus_{\mathfrak{f}|M|N} \bigoplus_{d|N/M} B(d) S_k^{\mathrm{new}}(\Gamma_0(M),\chi)\;.$$

Advantage: mostly canonical, elegant. Disadvantage: use of a slightly more complicated trace formula. Nonetheless in practice faster, even for computing the full cuspidal space.

Splitting and Computing Eigenforms

Once a basis of the newspace is obtained, we want to compute the eigenforms. This is simply linear algebra (essentially computing intersection of kernels), but note that the full Hecke algebra must be used, not just the individual $T(n)$. For instance, it is impossible to split $S_2^{\text{new}}(512)$ using $T(n)$'s alone (but together with e.g., $T(3)+T(5)$ splits).

Implementation bottleneck: character χ can have large order, so should work in large cyclotomic fields. In practice, work over finite fields.

Splitting and Computing Eigenforms

Once a basis of the newspace is obtained, we want to compute the eigenforms. This is simply linear algebra (essentially computing intersection of kernels), but note that the full Hecke algebra must be used, not just the individual $T(n)$. For instance, it is impossible to split $S_2^{\text{new}}(512)$ using $T(n)$'s alone (but together with e.g., $T(3)+T(5)$ splits).

Implementation bottleneck: character χ can have large order, so should work in large cyclotomic fields. In practice, work over finite fields.

Eisenstein Series I

Basis of Eisenstein series given by a theorem of J. Weisinger (slightly corrected). In weight $k \geq 3$, define for two primitive characters χ_1 modulo N_1 and χ_2 modulo N_2

$$F_k(\chi_1, \chi_2) = \delta_{N_2,1} \frac{L(\chi_1, 1-k)}{2} + \sum_{n \geq 1} \sigma_{k-1}(\chi_1, \chi_2; n) q^n ,$$

with

$$\sigma_{k-1}(\chi_1, \chi_2; n) = \sum_{d \mid n} \chi_1(d) \chi_2(n/d) d^{k-1} .$$

For $k \geq 3$, the $B(d)F_k(\chi_1, \chi_2)$ for $dN_1N_2 \mid N$ and $\chi \sim \chi_1\chi_2$ form a basis of the space of Eisenstein series in $M_k(\Gamma_0(N), \chi)$.

Eisenstein Series I

Basis of Eisenstein series given by a theorem of J. Weisinger (slightly corrected). In weight $k \geq 3$, define for two primitive characters χ_1 modulo N_1 and χ_2 modulo N_2

$$F_k(\chi_1, \chi_2) = \delta_{N_2,1} \frac{L(\chi_1, 1-k)}{2} + \sum_{n \geq 1} \sigma_{k-1}(\chi_1, \chi_2; n) q^n \;,$$

with

$$\sigma_{k-1}(\chi_1, \chi_2; n) = \sum_{d \mid n} \chi_1(d) \chi_2(n/d) d^{k-1} \;.$$

For $k \geq 3$, the $B(d) F_k(\chi_1, \chi_2)$ for $dN_1 N_2 \mid N$ and $\chi \sim \chi_1 \chi_2$ form a basis of the space of Eisenstein series in $M_k(\Gamma_0(N), \chi)$.

For $k = 1$ and $k = 2$ need to slightly modify:
For $k = 1$ exists a symmetry $F_1(\chi_2, \chi_1) = F_1(\chi_1, \chi_2)$ (the constant term in the above formula is slightly modified), so must restrict to only one among pairs (χ_1, χ_2) and (χ_2, χ_1), for instance by requiring χ_2 to be even (hence χ_1 odd).

For $k = 2$, the previous result is valid as is, except when χ is the trivial character. In that case, choose (χ_1, χ_2) as before, but exclude the case χ_1 and χ_2 trivial ($E_2(\tau)$ is not quite modular). In compensation, add the forms $E_2(\tau) - dE_2(d\tau)$ for all $d \mid N$, $d > 1$.

Eisenstein Series II

For $k = 1$ and $k = 2$ need to slightly modify:
For $k = 1$ exists a symmetry $F_1(\chi_2, \chi_1) = F_1(\chi_1, \chi_2)$ (the constant term in the above formula is slightly modified), so must restrict to only one among pairs (χ_1, χ_2) and (χ_2, χ_1), for instance by requiring χ_2 to be even (hence χ_1 odd).

For $k = 2$, the previous result is valid as is, except when χ is the trivial character. In that case, choose (χ_1, χ_2) as before, but exclude the case χ_1 and χ_2 trivial ($E_2(\tau)$ is not quite modular). In compensation, add the forms $E_2(\tau) - dE_2(d\tau)$ for all $d \mid N$, $d > 1$.

However, above construction not efficient since coefficients in the (usually large) number field $\mathbb{Q}(\chi_1, \chi_2)$. A generalization of Hilbert 90 due to P. Cartier tells us that we can go down to the (usually much smaller) field $\mathbb{Q}(\chi)$:

Theorem. Let k be a field, M a finite-dimensional $\overline{k}$-vector space, and assume given an action of $G = \mathrm{Gal}(\overline{k}/k)$ on M compatible with the $\overline{k}$-structure. For $k \subset K \subset \overline{k}$ we say that $\mu \in M$ is defined over K if μ is fixed by every element $\sigma \in \mathrm{Gal}(\overline{k}/K)$. Assume that every element of M can be defined over a finite extension of k. Then M has a $\overline{k}$-basis consisting of elements defined over the base field k.

Easy proof. Also, not difficult to apply explicitly to our case.

However, above construction not efficient since coefficients in the (usually large) number field $\mathbb{Q}(\chi_1, \chi_2)$. A generalization of Hilbert 90 due to P. Cartier tells us that we can go down to the (usually much smaller) field $\mathbb{Q}(\chi)$:

Theorem. Let k be a field, M a finite-dimensional $\overline{k}$-vector space, and assume given an action of $G = \mathrm{Gal}(\overline{k}/k)$ on M compatible with the $\overline{k}$-structure. For $k \subset K \subset \overline{k}$ we say that $\mu \in M$ is defined over K if μ is fixed by every element $\sigma \in \mathrm{Gal}(\overline{k}/K)$. Assume that every element of M can be defined over a finite extension of k. Then M has a $\overline{k}$-basis consisting of elements defined over the base field k.

Easy proof. Also, not difficult to apply explicitly to our case.

However, above construction not efficient since coefficients in the (usually large) number field $\mathbb{Q}(\chi_1, \chi_2)$. A generalization of Hilbert 90 due to P. Cartier tells us that we can go down to the (usually much smaller) field $\mathbb{Q}(\chi)$:

Theorem. Let k be a field, M a finite-dimensional $\overline{k}$-vector space, and assume given an action of $G = \mathrm{Gal}(\overline{k}/k)$ on M compatible with the $\overline{k}$-structure. For $k \subset K \subset \overline{k}$ we say that $\mu \in M$ is defined over K if μ is fixed by every element $\sigma \in \mathrm{Gal}(\overline{k}/K)$. Assume that every element of M can be defined over a finite extension of k. Then M has a $\overline{k}$-basis consisting of elements defined over the base field k.

Easy proof. Also, not difficult to apply explicitly to our case.

For many reasons, also need to compute values at all cusps of the basis of $B(d)F_k(\chi_1, \chi_2)$ (and their "Cartier projections"). Easy in weight $k \geq 3$, simple in weight $k = 2$, but much more intricate in weight $k = 1$ because of necessary analytic continuation.

Result in weight 1 of the form $f(\chi_1, \chi_2) + f(\chi_2, \chi_1)$ with an explicit function f depending on the cusp and the characters, necessarily of this form because of the symmetry $F_1(\chi_2, \chi_1) = F_1(\chi_1, \chi_2)$.

Eisenstein Series III

For many reasons, also need to compute values at all cusps of the basis of $B(d)F_k(\chi_1,\chi_2)$ (and their "Cartier projections"). Easy in weight $k\ge 3$, simple in weight $k=2$, but much more intricate in weight $k=1$ because of necessary analytic continuation.

Result in weight 1 of the form $f(\chi_1,\chi_2)+f(\chi_2,\chi_1)$ with an explicit function f depending on the cusp and the characters, necessarily of this form because of the symmetry $F_1(\chi_2,\chi_1)=F_1(\chi_1,\chi_2)$.

Modular Forms of Weight 1: The Full Cuspidal Space I

We do not claim at any originality, but weight 1 modular forms are available. Many methods available. All rely on computing quotients of modular forms of higher weight. The main problem is to determine when a form is divisible by another.

We use a method due to G. Schaeffer called Hecke Stability: if V is a finite-dimensional subspace of the infinite-dimensional space of modular functions (with poles) of weight 1 stable by some $T(p)$ with p not dividing the level, then $V \subset M_1(\Gamma_0(N), \chi)$.

Sketch of proof: if not, there would exist a pole of order at least 1 at some cusp, and the iterated action of $T(p)$ would make the pole of arbitrarily large order, contradicting the finite-dimensionality of V.

Need additional conditions to have first $S_1(\Gamma_0(N), \chi) \subset V$, and second $S_1(\Gamma_0(N), \chi) = V$. Depends on how V is created.

We do not claim at any originality, but weight 1 modular forms are available. Many methods available. All rely on computing quotients of modular forms of higher weight. The main problem is to determine when a form is divisible by another.

We use a method due to G. Schaeffer called Hecke Stability: if V is a finite-dimensional subspace of the infinite-dimensional space of modular functions (with poles) of weight 1 stable by some $T(p)$ with p not dividing the level, then $V \subset M_1(\Gamma_0(N), \chi)$.

Sketch of proof: if not, there would exist a pole of order at least 1 at some cusp, and the iterated action of $T(p)$ would make the pole of arbitrarily large order, contradicting the finite-dimensionality of V.

Need additional conditions to have first $S_1(\Gamma_0(N), \chi) \subset V$, and second $S_1(\Gamma_0(N), \chi) = V$. Depends on how V is created.

Modular Forms of Weight 1: The Full Cuspidal Space I

We do not claim at any originality, but weight 1 modular forms are available. Many methods available. All rely on computing quotients of modular forms of higher weight. The main problem is to determine when a form is divisible by another.

We use a method due to G. Schaeffer called Hecke Stability: if V is a finite-dimensional subspace of the infinite-dimensional space of modular functions (with poles) of weight 1 stable by some $T(p)$ with p not dividing the level, then $V \subset M_1(\Gamma_0(N), \chi)$.

Sketch of proof: if not, there would exist a pole of order at least 1 at some cusp, and the iterated action of $T(p)$ would make the pole of arbitrarily large order, contradicting the finite-dimensionality of V.

Need additional conditions to have first $S_1(\Gamma_0(N), \chi) \subset V$, and second $S_1(\Gamma_0(N), \chi) = V$. Depends on how V is created.

Modular Forms of Weight 1: The Full Cuspidal Space I

We do not claim at any originality, but weight 1 modular forms are available. Many methods available. All rely on computing quotients of modular forms of higher weight. The main problem is to determine when a form is divisible by another.

We use a method due to G. Schaeffer called Hecke Stability: if V is a finite-dimensional subspace of the infinite-dimensional space of modular functions (with poles) of weight 1 stable by some $T(p)$ with p not dividing the level, then $V \subset M_1(\Gamma_0(N), \chi)$.

Sketch of proof: if not, there would exist a pole of order at least 1 at some cusp, and the iterated action of $T(p)$ would make the pole of arbitrarily large order, contradicting the finite-dimensionality of V.

Need additional conditions to have first $S_1(\Gamma_0(N), \chi) \subset V$, and second $S_1(\Gamma_0(N), \chi) = V$. Depends on how V is created.

Modular Forms of Weight 1: The Full Cuspidal Space II

We will choose a set $\mathcal{E} \subset M_1(\Gamma_0(N), \overline{\chi})$ whose coefficients are known explicitly. The simplest is to take Eisenstein series. For every $E \in \mathcal{E}$ we define $V_E = S_2(\Gamma_0(N))/E$ and $V = \bigcap_{E \in \mathcal{E}} V_E$.

V is evidently a finite-dimensional subspace of modular functions of weight 1, level N, character χ, so we can apply Schaeffer's theorem. Furthermore $S_1(\Gamma_0(N), \chi) \subset V$, so the maximal subspace W of V stable by some $T(p)$ with $p \nmid N$ will satisfy

$$S_1(\Gamma_0(N), \chi) \subset W \subset M_1(\Gamma_0(N), \chi)\ .$$

Modular Forms of Weight 1: The Full Cuspidal Space II

We will choose a set $\mathcal{E} \subset M_1(\Gamma_0(N), \overline{\chi})$ whose coefficients are known explicitly. The simplest is to take Eisenstein series. For every $E \in \mathcal{E}$ we define $V_E = S_2(\Gamma_0(N))/E$ and $V = \bigcap_{E \in \mathcal{E}} V_E$.

V is evidently a finite-dimensional subspace of modular functions of weight 1, level N, character χ, so we can apply Schaeffer's theorem. Furthermore $S_1(\Gamma_0(N), \chi) \subset V$, so the maximal subspace W of V stable by some $T(p)$ with $p \nmid N$ will satisfy

$$S_1(\Gamma_0(N), \chi) \subset W \subset M_1(\Gamma_0(N), \chi)\ .$$

Modular Forms of Weight 1: The Full Cuspidal Space III

To obtain $S_1(\Gamma_0(N), \chi)$ exactly, need the following: for any cusp s, there must exist $E \in \mathcal{E}$ which does not vanish at s. Easy thanks to the computation of the values of Eisenstein series at cusps done above. If desired, using suitable linear combination can have $\mathcal{E}$ contain a single Eisenstein series.

Main implementation bottleneck: once again, values of χ may belong to a large cyclotomic field, so here absolutely essential to work in finite fields.

Modular Forms of Weight 1: The Full Cuspidal Space III

To obtain $S_1(\Gamma_0(N), \chi)$ exactly, need the following: for any cusp s, there must exist $E \in \mathcal{E}$ which does not vanish at s. Easy thanks to the computation of the values of Eisenstein series at cusps done above. If desired, using suitable linear combination can have $\mathcal{E}$ contain a single Eisenstein series.

Main implementation bottleneck: once again, values of χ may belong to a large cyclotomic field, so here absolutely essential to work in finite fields.

Modular Forms of Weight 1: The Full Cuspidal Space IV

Note that e.g. $S/E_1 \cap S/E_2$ is computed as follows: first we compute $SE_2 \cap SE_1$ as $S'E_2$, so $S/E_1 \cap S/E_2 = S'/E_1$. Thus, never really need to compute $1/E$ except at the end to obtain Fourier coefficients. If necessary, can compute it over finite fields.

A variant of this implementation gives modular forms of weight 1 modulo p which cannot be lifted to characteristic 0, so are genuine mod p objects. We have not (yet) implemented this variant.

Modular Forms of Weight 1: The Full Cuspidal Space IV

Note that e.g. $S/E_1 \cap S/E_2$ is computed as follows: first we compute $SE_2 \cap SE_1$ as $S'E_2$, so $S/E_1 \cap S/E_2 = S'/E_1$. Thus, never really need to compute $1/E$ except at the end to obtain Fourier coefficients. If necessary, can compute it over finite fields.

A variant of this implementation gives modular forms of weight 1 modulo p which cannot be lifted to characteristic 0, so are genuine mod p objects. We have not (yet) implemented this variant.

Modular Forms of Weight 1: The Newspace

First note that the dimension of the newspace can be easily obtained by a form of Möbius inversion. The newspace itself is more difficult since its initial definition involves analytic objects (e.g., Petersson products). Nonetheless, can obtain the newspace by purely algebraic means, as follows:

1. Compute the matrix of $T(n)$ on the given basis of S_1, then its trace, for a small number of values of n (up to the Sturm bound).
2. Using the "surprising theorem" above, identify the trace form $\mathcal{T}(N)$ as a linear combination elements of the basis from these few initial traces.
3. Using the formula linking $\mathrm{Tr}^{\mathrm{new}}$ and Tr, identify the new trace form $\mathcal{T}^{\mathrm{new}}(N)$.
4. Finish the construction of $S_1^{\mathrm{new}}(\Gamma_0(N), \chi)$ as usual by applying Hecke operators on $\mathcal{T}^{\mathrm{new}}(N)$.
5. Split and find the eigenforms.

Modular Forms of Weight 1: The Newspace

First note that the dimension of the newspace can be easily obtained by a form of Möbius inversion. The newspace itself is more difficult since its initial definition involves analytic objects (e.g., Petersson products). Nonetheless, can obtain the newspace by purely algebraic means, as follows:

1. Compute the matrix of $T(n)$ on the given basis of S_1, then its trace, for a small number of values of n (up to the Sturm bound).
2. Using the "surprising theorem" above, identify the trace form $\mathcal{T}(N)$ as a linear combination elements of the basis from these few initial traces.
3. Using the formula linking $\mathrm{Tr}^{\mathrm{new}}$ and Tr, identify the new trace form $\mathcal{T}^{\mathrm{new}}(N)$.
4. Finish the construction of $S_1^{\mathrm{new}}(\Gamma_0(N), \chi)$ as usual by applying Hecke operators on $\mathcal{T}^{\mathrm{new}}(N)$.
5. Split and find the eigenforms.

Modular Forms of Weight 1: The Newspace

First note that the dimension of the newspace can be easily obtained by a form of Möbius inversion. The newspace itself is more difficult since its initial definition involves analytic objects (e.g., Petersson products). Nonetheless, can obtain the newspace by purely algebraic means, as follows:

1. Compute the matrix of $T(n)$ on the given basis of S_1, then its trace, for a small number of values of n (up to the Sturm bound).
2. Using the "surprising theorem" above, identify the trace form $\mathcal{T}(N)$ as a linear combination elements of the basis from these few initial traces.
3. Using the formula linking $\mathrm{Tr}^{\mathrm{new}}$ and Tr, identify the new trace form $\mathcal{T}^{\mathrm{new}}(N)$.
4. Finish the construction of $S_1^{\mathrm{new}}(\Gamma_0(N),\chi)$ as usual by applying Hecke operators on $\mathcal{T}^{\mathrm{new}}(N)$.
5. Split and find the eigenforms.

Modular Forms of Weight 1: The Newspace

First note that the dimension of the newspace can be easily obtained by a form of Möbius inversion. The newspace itself is more difficult since its initial definition involves analytic objects (e.g., Petersson products). Nonetheless, can obtain the newspace by purely algebraic means, as follows:

1. Compute the matrix of $T(n)$ on the given basis of S_1, then its trace, for a small number of values of n (up to the Sturm bound).
2. Using the "surprising theorem" above, identify the trace form $\mathcal{T}(N)$ as a linear combination elements of the basis from these few initial traces.
3. Using the formula linking $\mathrm{Tr}^{\mathrm{new}}$ and Tr, identify the new trace form $\mathcal{T}^{\mathrm{new}}(N)$.
4. Finish the construction of $S_1^{\mathrm{new}}(\Gamma_0(N), \chi)$ as usual by applying Hecke operators on $\mathcal{T}^{\mathrm{new}}(N)$.
5. Split and find the eigenforms.

Modular Forms of Weight 1: The Newspace

First note that the dimension of the newspace can be easily obtained by a form of Möbius inversion. The newspace itself is more difficult since its initial definition involves analytic objects (e.g., Petersson products). Nonetheless, can obtain the newspace by purely algebraic means, as follows:

1. Compute the matrix of $T(n)$ on the given basis of S_1, then its trace, for a small number of values of n (up to the Sturm bound).
2. Using the "surprising theorem" above, identify the trace form $\mathcal{T}(N)$ as a linear combination elements of the basis from these few initial traces.
3. Using the formula linking $\mathrm{Tr}^{\mathrm{new}}$ and Tr, identify the new trace form $\mathcal{T}^{\mathrm{new}}(N)$.
4. Finish the construction of $S_1^{\mathrm{new}}(\Gamma_0(N), \chi)$ as usual by applying Hecke operators on $\mathcal{T}^{\mathrm{new}}(N)$.
5. Split and find the eigenforms.

Modular Forms of Weight 1: The Newspace

First note that the dimension of the newspace can be easily obtained by a form of Möbius inversion. The newspace itself is more difficult since its initial definition involves analytic objects (e.g., Petersson products). Nonetheless, can obtain the newspace by purely algebraic means, as follows:

1. Compute the matrix of $T(n)$ on the given basis of S_1, then its trace, for a small number of values of n (up to the Sturm bound).
2. Using the "surprising theorem" above, identify the trace form $\mathcal{T}(N)$ as a linear combination elements of the basis from these few initial traces.
3. Using the formula linking $\mathrm{Tr}^{\mathrm{new}}$ and Tr, identify the new trace form $\mathcal{T}^{\mathrm{new}}(N)$.
4. Finish the construction of $S_1^{\mathrm{new}}(\Gamma_0(N),\chi)$ as usual by applying Hecke operators on $\mathcal{T}^{\mathrm{new}}(N)$.
5. Split and find the eigenforms.

Modular Forms of Weight 1: Tables

A table of modular forms of weight 1 (up to level 1500, probably 2000 soon) has been computed by K. Buzzard and A. Lauder, available in Sage and magma format on Lauder's web page.

Using our implementation, it is easy to build a smaller table (say up to level 500), including the mfclosures (see below), which allow to compute as many coefficients as one likes.

Modular Forms of Weight 1: Tables

A table of modular forms of weight 1 (up to level 1500, probably 2000 soon) has been computed by K. Buzzard and A. Lauder, available in Sage and magma format on Lauder's web page.

Using our implementation, it is easy to build a smaller table (say up to level 500), including the mfclosures (see below), which allow to compute as many coefficients as one likes.

Part II: Implementation Details

Must explain how to represent:

- Dirichlet characters.
- values of Dirichlet characters.
- Modular Forms.
- Eigenforms.

In addition, brief description of available functions on these objects.

Part II: Implementation Details

Must explain how to represent:

- Dirichlet characters.
- values of Dirichlet characters.
- Modular Forms.
- Eigenforms.

In addition, brief description of available functions on these objects.

Dirichlet Characters I

Most commonly used method to represent Characters is now the Conrey representation, due to B. Conrey, which gives a semi-canonical isomorphism between $(\mathbb{Z}/N\mathbb{Z})^*$ and its group of characters.

In the present `Pari/GP` implementation (but this may change) a Dirichlet character is represented by a triple (G, C, o), where G is the `Pari` representation of the group $(\mathbb{Z}/N\mathbb{Z})^*$, C is the Conrey representation, and o is the order of the character (it can of course be deduced from C, but it is useful to carry it along). Such a triple will be called an mfcharacter.

Main creation function mfcharcreate. Examples:

`mfcharcreate(Mod(5,96))`: 5th Conrey char modulo 96.

`mfcharcreate(-163)`: Kronecker symbol $\left(\frac{-163}{n}\right)$.

Dirichlet Characters I

Most commonly used method to represent Characters is now the Conrey representation, due to B. Conrey, which gives a semi-canonical isomorphism between $(\mathbb{Z}/N\mathbb{Z})^*$ and its group of characters.

In the present `Pari/GP` implementation (but this may change) a Dirichlet character is represented by a triple (G, C, o), where G is the `Pari` representation of the group $(\mathbb{Z}/N\mathbb{Z})^*$, C is the Conrey representation, and o is the order of the character (it can of course be deduced from C, but it is useful to carry it along). Such a triple will be called an mfcharacter.

Main creation function mfcharcreate. Examples:

`mfcharcreate(Mod(5,96))`: 5th Conrey char modulo 96.

`mfcharcreate(-163)`: Kronecker symbol $\left(\frac{-163}{n}\right)$.

Dirichlet Characters I

Most commonly used method to represent Characters is now the Conrey representation, due to B. Conrey, which gives a semi-canonical isomorphism between $(\mathbb{Z}/N\mathbb{Z})^*$ and its group of characters.
In the present `Pari/GP` implementation (but this may change) a Dirichlet character is represented by a triple (G, C, o), where G is the `Pari` representation of the group $(\mathbb{Z}/N\mathbb{Z})^*$, C is the Conrey representation, and o is the order of the character (it can of course be deduced from C, but it is useful to carry it along). Such a triple will be called an mfcharacter.

Main creation function mfcharcreate. Examples:
`mfcharcreate(Mod(5,96))`: 5th Conrey char modulo 96.
`mfcharcreate(-163)`: Kronecker symbol $\left(\dfrac{-163}{n}\right)$.

Dirichlet Characters II

All usual functions to handle mfcharacters: multiplication, division, conjugation, conductor, reduction to primitive character, etc... all of the form mfcharxxx.

Evaluation of an mfcharacter: can evaluate as a complex number `mfcharcxeval`, but much more often as an algebraic number, in `Pari/GP` a `polmod`, such as `Mod(t,t^2+t+1)`, function `mfchareval`.
The default variable is `t`, but it is preferable to have the user choose it.

Dirichlet Characters II

All usual functions to handle mfcharacters: multiplication, division, conjugation, conductor, reduction to primitive character, etc... all of the form mfcharxxx.

Evaluation of an mfcharacter: can evaluate as a complex number `mfcharcxeval`, but much more often as an algebraic number, in `Pari/GP` a `polmod`, such as `Mod(t,t^2+t+1)`, function `mfchareval`.
The default variable is `t`, but it is preferable to have the user choose it.

Representation of Modular Forms I

In the modular symbol context, modular forms are represented by modular symbols, which are finite objects. In the trace formula context, the situation is different. We cannot represent them by their Fourier expansion at infinity since this is an infinite object. We have chosen to do as follows:

A modular form will be represented as an mfclosure (our terminology), which is a `Pari/GP` object on which the function mfan can be applied (as well as functions derived from it): `mfan(F,n)` gives the vector of Fourier coefficients $[a(0), a(1), \ldots, a(n)]$ (warning: begins at $a(0)$). To have a power series expansion, use the `GP` function `Ser`; example: `Ser(mfan(mfdeltaclos(),5),q)` returns

$$q - 24q^2 + 252q^3 - 1472q^4 + 4830q^5 + O(q^6)$$

Representation of Modular Forms I

In the modular symbol context, modular forms are represented by modular symbols, which are finite objects. In the trace formula context, the situation is different. We cannot represent them by their Fourier expansion at infinity since this is an infinite object. We have chosen to do as follows:

A modular form will be represented as an mfclosure (our terminology), which is a `Pari/GP` object on which the function mfan can be applied (as well as functions derived from it): `mfan(F,n)` gives the vector of Fourier coefficients $[a(0), a(1), \dots, a(n)]$ (warning: begins at $a(0)$). To have a power series expansion, use the `GP` function `Ser`; example: `Ser(mfan(mfdeltaclos(),5),q)` returns

$$q - 24q^2 + 252q^3 - 1472q^4 + 4830q^5 + O(q^6)$$

Representation of Modular Forms II

Advantage of this representation: no need to specify in advance the desired number of Fourier coefficients.
All the usual arithmetic and modular form operations on mfclosures are of course available.

Note that the internal representation will be in direct Polish notation. For instance, the modular form $E_4(\Delta^2+E_{24})$ is represented (with evident notation) by

```
Mul(E(4),Add(Mul(Delta(),Delta()),E(24)))
```

Representation of Modular Forms II

Advantage of this representation: no need to specify in advance the desired number of Fourier coefficients.
All the usual arithmetic and modular form operations on mfclosures are of course available.

Note that the internal representation will be in direct Polish notation. For instance, the modular form $E_4(\Delta^2 + E_{24})$ is represented (with evident notation) by

```
Mul(E(4),Add(Mul(Delta(),Delta()),E(24)))
```

Other Functions I

In addition to the standard operations, several operations involve numerical work:

- Evaluation of a modular form at a point in $\mathbb{H}$.
- Computation of values (including special values) of the L-function and symmetric square L-function attaced to a modular form, hence of the Petersson square.
- More difficult: numerical evaluation (and algebraic recognition) of eigenvalues of Atkin–Lehner operators and the Fricke involution when the explicit formulas are not available.
- Computing the Fourier expansion at cusps other than ∞: exactly as in the case of modular symbols, this can easily be done if the cusp can be attained from ∞ by the Atkin–Lehner operators, otherwise it is much more difficult to find this expansion.

Other Functions II

Using the formulas of N. Skoruppa and D. Zagier, in the case of trivial character (Haupttypus) we can generate automatically the Atkin–Lehner spaces, which in turn need to be split to find the eigenforms.

Also, linear decomposition of a given form on basis (with or without Eisenstein series).

There is also a search function for finding rational eigenforms with given initial coefficients. I have available a (large) table containing all rational eigenforms of level $N \leq 300$ and weight $1 \leq k \leq 14$, necessarily with trivial or quadratic character, which can easily be extended if necessary.

Other Functions II

Using the formulas of N. Skoruppa and D. Zagier, in the case of trivial character (Haupttypus) we can generate automatically the Atkin–Lehner spaces, which in turn need to be split to find the eigenforms.
Also, linear decomposition of a given form on basis (with or without Eisenstein series).

There is also a search function for finding rational eigenforms with given initial coefficients. I have available a (large) table containing all rational eigenforms of level $N \leq 300$ and weight $1 \leq k \leq 14$, necessarily with trivial or quadratic character, which can easily be extended if necessary.

Other Functions II

Using the formulas of N. Skoruppa and D. Zagier, in the case of trivial character (Haupttypus) we can generate automatically the Atkin–Lehner spaces, which in turn need to be split to find the eigenforms.
Also, linear decomposition of a given form on basis (with or without Eisenstein series).

There is also a search function for finding rational eigenforms with given initial coefficients. I have available a (large) table containing all rational eigenforms of level $N \leq 300$ and weight $1 \leq k \leq 14$, necessarily with trivial or quadratic character, which can easily be extended if necessary.

Taylor Series Expansions I

In addition, preliminary implementation of Taylor series expansions of modular forms, for now only for the full modular group around $\tau = i$. As with modular symbols, Taylor series expansions involve only a finite amount of data: given a few initial coefficients, all the others can be obtained using a pseudo-recursion, contrary to Fourier expansions.

Typically a pseudo-recursion for a sequence u_n is a true recursion on polynomials in one variable $P_n(X)$ such that $u_n = P_n(0)$.

Remark: Taylor series expansions are essential when working in cocompact subgroups, since in that case there are no cusps.

Taylor Series Expansions I

In addition, preliminary implementation of Taylor series expansions of modular forms, for now only for the full modular group around $\tau = i$. As with modular symbols, Taylor series expansions involve only a finite amount of data: given a few initial coefficients, all the others can be obtained using a pseudo-recursion, contrary to Fourier expansions.

Typically a pseudo-recursion for a sequence u_n is a true recursion on polynomials in one variable $P_n(X)$ such that $u_n = P_n(0)$.

Remark: Taylor series expansions are essential when working in cocompact subgroups, since in that case there are no cusps.

Taylor Series Expansions I

In addition, preliminary implementation of Taylor series expansions of modular forms, for now only for the full modular group around $\tau = i$. As with modular symbols, Taylor series expansions involve only a finite amount of data: given a few initial coefficients, all the others can be obtained using a pseudo-recursion, contrary to Fourier expansions.

Typically a pseudo-recursion for a sequence u_n is a true recursion on polynomials in one variable $P_n(X)$ such that $u_n = P_n(0)$.

Remark: Taylor series expansions are essential when working in cocompact subgroups, since in that case there are no cusps.

Example for the Ramanujan Δ-function around $\tau = i$:

$P_{-1}(X) = 0$, $P_0(X) = 1 - X^2$, and for $n \geq 0$

$$P_{n+1}(X) = -\frac{n+6}{6} X P_n(X) + \frac{X^2-1}{2} P_n'(X) - \frac{n(n+11)}{144} P_{n-1}(X)\;.$$

$$\Delta(\tau) = \frac{2^{12}}{(\tau+i)^{12}} \sum_{n\geq 0} \frac{C_n}{n!} \left(\frac{\tau - i}{\tau + i}\right)^n\;,$$

where $C_n = A^6 \cdot B^n \cdot P_n(0)$, with A and B easily computable universal constants (for instance $A = \Gamma(1/4)^4/(16\pi^3)$).

Taylor Series Expansions II

Example for the Ramanujan Δ-function around $\tau = i$:

$P_{-1}(X) = 0$, $P_0(X) = 1 - X^2$, and for $n \geq 0$

$$P_{n+1}(X) = -\frac{n+6}{6} X P_n(X) + \frac{X^2 - 1}{2} P_n'(X) - \frac{n(n+11)}{144} P_{n-1}(X) .$$

$$\Delta(\tau) = \frac{2^{12}}{(\tau + i)^{12}} \sum_{n \geq 0} \frac{C_n}{n!} \left(\frac{\tau - i}{\tau + i} \right)^n ,$$

where $C_n = A^6 \cdot B^n \cdot P_n(0)$, with A and B easily computable universal constants (for instance $A = \Gamma(1/4)^4/(16\pi^3)$).

The basic initialization command for spaces of cusp forms is mfinit, with optional character and space either new or full.

```
? mf=mfinit([26,2]);
```

Creates a basis of the newspace of level 26 weight 2, no character.

```
? L=mfgetclos(mf);mfclostoser(L,10,q)
```

get the corresponding mfclosures; there are two:

$[2q-2q^3+2q^4-4q^5-\ldots,-2q-4q^2+10q^3-2q^4+\ldots]$

These are of course not the eigenforms. To get them must split:

```
? LE=mfgeteigenclos(mfsplit(mf));mfclostoser(LE,10,q)
```

$[q-q^2+q^3+q^4-3q^5-\ldots,q+q^2-3q^3+q^4-q^5-\ldots]$

Part III: Example Commands I

The basic initialization command for spaces of cusp forms is mfinit, with optional character and space either new or full.

```
?  mf=mfinit([26,2]);
```

Creates a basis of the newspace of level 26 weight 2, no character.

`?  L=mfgetclos(mf);mfclostoser(L,10,q)` get the corresponding mfclosures; there are two:

$[2q - 2q^3 + 2q^4 - 4q^5 - ..., -2q - 4q^2 + 10q^3 - 2q^4 + ...]$

These are of course not the eigenforms. To get them must split:

```
?
LE=mfgeteigenclos(mfsplit(mf));mfclostoser(LE,10,q)
```

$[q - q^2 + q^3 + q^4 - 3q^5 - ..., q + q^2 - 3q^3 + q^4 - q^5 - ...]$

Part III: Example Commands I

The basic initialization command for spaces of cusp forms is mfinit, with optional character and space either new or full.

```
? mf=mfinit([26,2]);
```

Creates a basis of the newspace of level 26 weight 2, no character.

```
? L=mfgetclos(mf);mfclostoser(L,10,q)
```

get the corresponding mfclosures; there are two:

$[2q - 2q^3 + 2q^4 - 4q^5 - ..., -2q - 4q^2 + 10q^3 - 2q^4 + ...]$

These are of course not the eigenforms. To get them must split:

```
?
LE=mfgeteigenclos(mfsplit(mf));mfclostoser(LE,10,q)
```

$[q - q^2 + q^3 + q^4 - 3q^5 - ..., q + q^2 - 3q^3 + q^4 - q^5 - ...]$

Part III: Example Commands II

Splitting can be over a number field:

```
?  mf=mfsplit(mfinit([23,2]));LE=mfgeteigenclos(mf);
?  mfclostoser(LE,10,q)
```

$[Mod(1, y^2 + y - 1)q + Mod(y, y^2 + y - 1)q^2 + ...]$

To see it better:

```
[mfgalpols(mf),lift(mfclostoser(LE,10,q))]
```

$[[y^2 + y - 1], [q + yq^2 + (-2y - 1)q^3 + (-y - 1)q^4 + 2yq^5 ...]]$

When there are characters: function `mfcharcreate` either with a fundamental discriminant D, an intmod $Mod(m, N)$: m-th Conrey character modulo N, or $[ZN, \chi]$ standard GP character.

`CHI=mfcharcreate(Mod(2,5));`

`vector(5,n,mfchareval(CHI,n))` (or simply `mfchartovec(CHI)`)

$[1, Mod(t, t^2 + 1), Mod(-t, t^2 + 1), -1, 0]$

Note on variables: q can be used for modular form expansions, y for number fields occurring in splittings, t for characters. The variable y is compulsory, the others at the user's choice.

Part III: Example Commands II

Splitting can be over a number field:

```
?  mf=mfsplit(mfinit([23,2]));LE=mfgeteigenclos(mf);
?  mfclostoser(LE,10,q)
```

$[Mod(1,y^2+y-1)q+Mod(y,y^2+y-1)q^2+...]$

To see it better:

```
[mfgalpols(mf),lift(mfclostoser(LE,10,q))]
```

$[[y^2+y-1],[q+yq^2+(-2y-1)q^3+(-y-1)q^4+2yq^5...]]$

When there are characters: function `mfcharcreate` either with a fundamental discriminant D, an intmod $Mod(m,N)$: m-th Conrey character modulo N, or $[ZN,\chi]$ standard GP character.

`CHI=mfcharcreate(Mod(2,5));`

`vector(5,n,mfchareval(CHI,n))` (or simply `mfchartovec(CHI)`)

$[1,Mod(t,t^2+1),Mod(-t,t^2+1),-1,0]$

Note on variables: q can be used for modular form expansions, y for number fields occurring in splittings, t for characters. The variable y is compulsory, the others at the user's choice.

Part III: Example Commands II

Splitting can be over a number field:

```
?  mf=mfsplit(mfinit([23,2]));LE=mfgeteigenclos(mf);
?  mfclostoser(LE,10,q)
```

$[Mod(1, y^2 + y - 1)q + Mod(y, y^2 + y - 1)q^2 + ...]$

To see it better:

```
[mfgalpols(mf),lift(mfclostoser(LE,10,q))]
```

$[[y^2 + y - 1], [q + yq^2 + (-2y - 1)q^3 + (-y - 1)q^4 + 2yq^5 ...]]$

When there are characters: function `mfcharcreate` either with a fundamental discriminant D, an intmod $Mod(m, N)$: m-th Conrey character modulo N, or $[ZN, \chi]$ standard GP character.

`CHI=mfcharcreate(Mod(2,5));`

`vector(5,n,mfchareval(CHI,n))` (or simply `mfchartovec(CHI)`)

$[1, Mod(t, t^2 + 1), Mod(-t, t^2 + 1), -1, 0]$

Note on variables: q can be used for modular form expansions, y for number fields occurring in splittings, t for characters. The variable y is compulsory, the others at the user's choice.

Example Commands III

Can get complicated but unavoidable:

```
mf=mfsplit(mfinit([15,3,CHI]));mfgalpols(mf)
```

$[y^2 + Mod(-3t, t^2 + 1)]$

Quadratic extension of quadratic extension.

```
lift(lift(mfclostoser(mfgeteigenclos(mf,10,q))))
```

$[q + (y - t - 1)q^2 + tyq^3 + ((-2t - 2)y + t)q^4 + ...]$

Can get expansion over an absolute instead of relative number field if desired:

```
mfreltoabs(mfclostoser(mfgeteigenclos(mf,10,q)))
```

$[Mod(1, y^4 + 9)q + Mod(-1/3y^2 + y - 1, y^4 + 9)q^2 + ...]$

Example Commands III

Can get complicated but unavoidable:

```
mf=mfsplit(mfinit([15,3,CHI]));mfgalpols(mf)
```

$[y^2 + Mod(-3t, t^2 + 1)]$

Quadratic extension of quadratic extension.

```
lift(lift(mfclostoser(mfgeteigenclos(mf,10,q))))
```

$[q + (y - t - 1)q^2 + tyq^3 + ((-2t - 2)y + t)q^4 + ...]$

Can get expansion over an absolute instead of relative number field if desired:

```
mfreltoabs(mfclostoser(mfgeteigenclos(mf,10,q)))
```

$[Mod(1, y^4 + 9)q + Mod(-1/3y^2 + y - 1, y^4 + 9)q^2 + ...]$

Example Commands IV

Can ask only rational spaces:

```
mf=mfsplit(mfinit([35,2]));mfgalpols(mf)
```

$[y, y^2 + y - 4]$

```
mf=mfsplit(mfinit([35,2]),1);mfclostoser(mfgeteigenclos(mf)
```

$[q + q^3 - 2q^4 - q^5 + \dots]$

This is in fact used by the function `mfsearch`:

```
S=mfsearch(40,[1,-2])
```

$[[34, 2, [\mathrm{Vecsmall}(\dots)]]]$

This tells us that there is only one rational eigenform such that $Nk \leq 80$, $a_2 = 1$, $a_3 = -2$, and it is in $S_2(\Gamma_0(34))$, and the beginning of its Fourier expansion is given by

```
mfclostoser(S[1][3],10,q)
```

$q + q^2 - 2q^3 + q^4 - 2q^6 - 4q^7 + \dots$

Note the advantage of using mfclosures: if we want the Fourier expansion to 100 terms, say, simply type `mfclostoser(S[1][3],100,q)`.

Example Commands IV

Can ask only rational spaces:

```
mf=mfsplit(mfinit([35,2]));mfgalpols(mf)
```

$[y, y^2 + y - 4]$

```
mf=mfsplit(mfinit([35,2]),1);mfclostoser(mfgeteigenclos(mf)
```

$[q + q^3 - 2q^4 - q^5 + ...]$

This is in fact used by the function `mfsearch`:

```
S=mfsearch(40,[1,-2])
```

$[[34, 2, [\text{Vecsmall}(...)]]]$

This tells us that there is only one rational eigenform such that $Nk \leq 80$, $a_2 = 1$, $a_3 = -2$, and it is in $S_2(\Gamma_0(34))$, and the beginning of its Fourier expansion is given by

```
mfclostoser(S[1][3],10,q)
```

$q + q^2 - 2q^3 + q^4 - 2q^6 - 4q^7 + ...$

Note the advantage of using mfclosures: if we want the Fourier expansion to 100 terms, say, simply type `mfclostoser(S[1][3],100,q)`.

Example Commands IV

Can ask only rational spaces:

`mf=mfsplit(mfinit([35,2]));mfgalpols(mf)`

$[y, y^2 + y - 4]$

`mf=mfsplit(mfinit([35,2]),1);mfclostoser(mfgeteigenclos(mf)`

$[q + q^3 - 2q^4 - q^5 + ...]$

This is in fact used by the function `mfsearch`:

`S=mfsearch(40,[1,-2])`

$[[34, 2, [\mathrm{Vecsmall}(...)]]]$

This tells us that there is only one rational eigenform such that $Nk \leq 80$, $a_2 = 1$, $a_3 = -2$, and it is in $S_2(\Gamma_0(34))$, and the beginning of its Fourier expansion is given by

`mfclostoser(S[1][3],10,q)`

$q + q^2 - 2q^3 + q^4 - 2q^6 - 4q^7 + ...$

Note the advantage of using mfclosures: if we want the Fourier expansion to 100 terms, say, simply type

`mfclostoser(S[1][3],100,q)`.

Example Commands V

Standard modular form functions: Hecke operators, $B(d)$ (i.e. expanding) operator, Atkin–Lehner operators, twist, and standard mfclosures (Eisenstein series E_k, Δ, j, etc...).

```
mf=mfinit([96,4]);M=mfmathecke(mf,5)
```

$[0,64,0,0,-84,0;1,-24/5,0,0,294/5,0;0,0,-30,-20,0,100,...]$

```
factor(charpoly(M))
```

$[x-10,2;x-2,2;x+14,2]$

```
M=mfmatatkin(mf,3)
```

$[1,[0,0,0,-24,0,-60;0,0,-9/5,-6/5,0,6;...]]$

```
mf2=mfsplit(mf);mfatkin(mf2,3)
```

$[[-1],[1],[-1],[1],[1],[-1]]$

Example Commands V

Standard modular form functions: Hecke operators, $B(d)$ (i.e. expanding) operator, Atkin–Lehner operators, twist, and standard mfclosures (Eisenstein series E_k, Δ, j, etc...).

```
mf=mfinit([96,4]);M=mfmathecke(mf,5)
```

$[0, 64, 0, 0, -84, 0; 1, -24/5, 0, 0, 294/5, 0; 0, 0, -30, -20, 0, 100, \ldots]$

```
factor(charpoly(M))
```

$[x - 10, 2; x - 2, 2; x + 14, 2]$

```
M=mfmatatkin(mf,3)
```

$[1, [0, 0, 0, -24, 0, -60; 0, 0, -9/5, -6/5, 0, 6; \ldots]]$

```
mf2=mfsplit(mf);mfatkin(mf2,3)
```

$[[-1], [1], [-1], [1], [1], [-1]]$

Standard modular form functions: Hecke operators, $B(d)$ (i.e. expanding) operator, Atkin–Lehner operators, twist, and standard mfclosures (Eisenstein series E_k, Δ, j, etc...).

```
mf=mfinit([96,4]);M=mfmathecke(mf,5)
```

$[0, 64, 0, 0, -84, 0; 1, -24/5, 0, 0, 294/5, 0; 0, 0, -30, -20, 0, 100, ...]$

```
factor(charpoly(M))
```

$[x - 10, 2; x - 2, 2; x + 14, 2]$

```
M=mfmatatkin(mf,3)
```

$[1, [0, 0, 0, -24, 0, -60; 0, 0, -9/5, -6/5, 0, 6; ...]]$

```
mf2=mfsplit(mf);mfatkin(mf2,3)
```

$[[-1], [1], [-1], [1], [1], [-1]]$

Example Commands V

Standard modular form functions: Hecke operators, $B(d)$ (i.e. expanding) operator, Atkin–Lehner operators, twist, and standard mfclosures (Eisenstein series E_k, Δ, j, etc...).

```
mf=mfinit([96,4]);M=mfmathecke(mf,5)
```

$[0, 64, 0, 0, -84, 0; 1, -24/5, 0, 0, 294/5, 0; 0, 0, -30, -20, 0, 100, ...]$

```
factor(charpoly(M))
```

$[x - 10, 2; x - 2, 2; x + 14, 2]$

```
M=mfmatatkin(mf,3)
```

$[1, [0, 0, 0, -24, 0, -60; 0, 0, -9/5, -6/5, 0, 6; ...]]$

```
mf2=mfsplit(mf);mfatkin(mf2,3)
```

$[[-1], [1], [-1], [1], [1], [-1]]$

Linear decompositions:

```
Th=1+2*sum(n=1,8,q^(n^2),O(q^80));
mf=mfinit([4,2],1);mfdecompose(mf,Th^4,1)
```

$[8, 16]$,

and if CHI is the nontrivial character modulo 4 (created for instance by `mfcharcreate(Mod(3,4))`)

```
mf=mfinit([4,5,CHI],1);mfdecompose(mf,Th^10,1)
```

$[64/5, 4/5, 32/5]$

Example Commands VI

Linear decompositions:

```
Th=1+2*sum(n=1,8,q^(n^2),O(q^80));
mf=mfinit([4,2],1);mfdecompose(mf,Th^4,1)
```

$[8, 16]$,

and if CHI is the nontrivial character modulo 4 (created for instance by `mfcharcreate(Mod(3,4))`)

```
mf=mfinit([4,5,CHI],1);mfdecompose(mf,Th^10,1)
```

$[64/5, 4/5, 32/5]$

Examples Commands VII

Eisenstein series:

```
CHI=mfcharcreate(Mod(3,4));L=mfeisenbasisclos([4,3,CHI]);
apply(E->mfclostoser(E,10,q),L)
```

$[q+4q^2+8q^3+16q^4+26q^5+..., -1/4+q+q^2-8q^3+q^4+26q^5-...]$

Numerical modular form functions such as evaluation at some point in the upper-half plane, computation of L-function values.

```
E4=mfEkclos(4);mfcloseval([1,4],E4,I)
```

1.4557628922687093224624220035988 69..

```
3*gamma(1/4)^8/(2*Pi)^6
```

1.4557628922687093224624220035988 69..

This is a consequence of complex multiplication.

Examples Commands VII

Eisenstein series:

```
CHI=mfcharcreate(Mod(3,4));L=mfeisenbasisclos([4,3,CHI]);
apply(E->mfclostoser(E,10,q),L)
```

$[q + 4q^2 + 8q^3 + 16q^4 + 26q^5 + ..., -1/4 + q + q^2 - 8q^3 + q^4 + 26q^5 - ...]$

Numerical modular form functions such as evaluation at some point in the upper-half plane, computation of L-function values.

```
E4=mfEkclos(4);mfcloseval([1,4],E4,I)
```

1.455762892268709322462422003598869..

```
3*gamma(1/4)^8/(2*Pi)^6
```

1.455762892268709322462422003598869..

This is a consequence of complex multiplication.

Example Commands VIII

Fourier series expansions at other cusps, under suitable assumptions:

```
mf=mfinit([5,8]);F=mfgetclos(mf)[1];mfclostoser(F,10,q)
```

$3q + 6q^2 - 28q^3 + 164q^4 - \ldots$

```
G=mfcuspexpansion(mf,F,0);mfclostoser(G,10,q)
```

$q + 34q^2 + 68q^3 + 28q^4 - \ldots$

Searching for modular eigenforms with given coefficients:

```
L=mfsearch(60,[-1,-3])
```

$[[53, 2, [\text{Vecsmall}\ldots]], [58, 2, [\text{Vecsmall}\ldots]]]$

```
[mfclostoser(L[1][3],10,q),mfclostoser(L[2][3],10,q)]
```

$[q - q^2 - 3q^3 - q^4 + 3q^6 - 4q^7 + \ldots, q - q^2 - 3q^3 + q^4 - 3q^5 + 3q^6 - 2q^7 \ldots]$

Example Commands VIII

Fourier series expansions at other cusps, under suitable assumptions:

```
mf=mfinit([5,8]);F=mfgetclos(mf)[1];mfclostoser(F,10,q)
```

$3q + 6q^2 - 28q^3 + 164q^4 - \dots$

```
G=mfcuspexpansion(mf,F,0);mfclostoser(G,10,q)
```

$q + 34q^2 + 68q^3 + 28q^4 - \dots$

Searching for modular eigenforms with given coefficients:

```
L=mfsearch(60,[-1,-3])
```

$[[53, 2, [\text{Vecsmall}\dots]], [58, 2, [\text{Vecsmall}\dots]]]$

```
[mfclostoser(L[1][3],10,q),mfclostoser(L[2][3],10,q)]
```

$[q - q^2 - 3q^3 - q^4 + 3q^6 - 4q^7 + \dots, q - q^2 - 3q^3 + q^4 - 3q^5 + 3q^6 - 2q^7\dots]$

Example Commands VIII

Fourier series expansions at other cusps, under suitable assumptions:

```
mf=mfinit([5,8]);F=mfgetclos(mf)[1];mfclostoser(F,10,q)
```

$3q + 6q^2 - 28q^3 + 164q^4 - ...$

```
G=mfcuspexpansion(mf,F,0);mfclostoser(G,10,q)
```

$q + 34q^2 + 68q^3 + 28q^4 - ...$

Searching for modular eigenforms with given coefficients:

```
L=mfsearch(60,[-1,-3])
```

$[[53, 2, [\text{Vecsmall}...]], [58, 2, [\text{Vecsmall}...]]]$

```
[mfclostoser(L[1][3],10,q),mfclostoser(L[2][3],10,q)]
```

$[q - q^2 - 3q^3 - q^4 + 3q^6 - 4q^7 + ..., q - q^2 - 3q^3 + q^4 - 3q^5 + 3q^6 - 2q^7 ...]$

Example Commands IX

Weight 1 computations:

L=mfchargalois(148,148,-1);

Finds all Galois equivalence classes of odd characters.

for(i=1,#L,print1(mfdim([148,1,L[i]])," "))

0 0 1 0 0 1 0 1 0

Three characters for which there exists a form of weight 1.

mf=mfinit([148,1,L[8]]); mfcoefs(mf[2][1],10,q)

$$Mod(1,t^2+1)*q+Mod(-t,t^2+1)*q^3+Mod(-1,t^2+1)*q^7+\ldots$$

mf2=mfsplit(mf); mfgaloistype(mf2)

Vecsmall([24]) Thus, this eigenform is of type S_4 (the lowest possible level).

mfgaloistype([675,1,mfcharcreate(Mod(161,675))])

Vecsmall([60]) This eigenform is of type A_5 (the lowest possible level is 633 but takes much longer to compute because of the order of the character).

Example Commands IX

Weight 1 computations:

L=mfchargalois(148,148,-1);

Finds all Galois equivalence classes of odd characters.

for(i=1,#L,print1(mfdim([148,1,L[i]])," "))

0 0 1 0 0 1 0 1 0

Three characters for which there exists a form of weight 1.

mf=mfinit([148,1,L[8]]); mfcoefstoser(mf[2][1],10,q)

$$Mod(1,t^2+1)*q+Mod(-t,t^2+1)*q^3+Mod(-1,t^2+1)*q^7+\dots$$

mf2=mfsplit(mf); mfgaloistype(mf2)

Vecsmall([24]) Thus, this eigenform is of type S_4 (the lowest possible level).

mfgaloistype([675,1,mfcharcreate(Mod(161,675))])

Vecsmall([60]) This eigenform is of type A_5 (the lowest possible level is 633 but takes much longer to compute because of the order of the character).

Example Commands IX

Weight 1 computations:

L=mfchargalois(148,148,-1);

Finds all Galois equivalence classes of odd characters.

for(i=1,#L,print1(mfdim([148,1,L[i]]),” ”))

0 0 1 0 0 1 0 1 0

Three characters for which there exists a form of weight 1.

mf=mfinit([148,1,L[8]]); mfclostoser(mf[2][1],10,q)

$$Mod(1,t^2+1)*q+Mod(-t,t^2+1)*q^3+Mod(-1,t^2+1)*q^7+\dots$$

mf2=mfsplit(mf); mfgaloistype(mf2)

Vecsmall([24]) Thus, this eigenform is of type S_4 (the lowest possible level).

mfgaloistype([675,1,mfcharcreate(Mod(161,675))])

Vecsmall([60]) This eigenform is of type A_5 (the lowest possible level is 633 but takes much longer to compute because of the order of the character).

Example Commands X

More typical examples:

mf=mfinit([148,1,L[3]]); mfgaloistype(mf)

Vecsmall([1800]) This eigenform is of type D_n with $n = 18$.

mfgaloistype([239,1,mfcharcreate(-239)])

Vecsmall([600, 1000, 3000])

Here there are three eigenforms, of type D_n with $n = 6$, 10, and 30 respectively.

Example Commands X

More typical examples:

```
mf=mfinit([148,1,L[3]]); mfgaloistype(mf)
```

Vecsmall([1800]) This eigenform is of type D_n with $n = 18$.

```
mfgaloistype([239,1,mfcharcreate(-239)])
```

Vecsmall([600, 1000, 3000])

Here there are three eigenforms, of type D_n with $n = 6$, 10, and 30 respectively.

Conclusion

As mentioned, contrary to modular symbols, trace formulas are less flexible for characterizing modular forms. The notion of mfclosure is a good, but not perfect, solution, since it allows to have the Fourier expansion to any (reasonable) desired number of terms, and not a fixed one. Possible alternative approaches are to use the known Fourier coefficients to compute either the corresponding modular symbol, or the Taylor expansion around $\tau = i$, which involves only a finite amount of data.

Thank you for your attention !

Conclusion

As mentioned, contrary to modular symbols, trace formulas are less flexible for characterizing modular forms. The notion of mfclosure is a good, but not perfect, solution, since it allows to have the Fourier expansion to any (reasonable) desired number of terms, and not a fixed one. Possible alternative approaches are to use the known Fourier coefficients to compute either the corresponding modular symbol, or the Taylor expansion around $\tau = i$, which involves only a finite amount of data.

Thank you for your attention !