# EHMM Calculator Documentation

A program created by M. Allen, B. Grove, L. Long, E. Rosen, and F.T. Tu
Documentation written by Esme Rosen*

## Contents



The EHMM calculator was developed to create demonstrations for the Explicit Hypergeometric Modularity method, introduced in [1] and [2], and studied further in [7], [12], [13], [14]. However, many features, especially the character sums, may be broadly applicable. In this file, we will explain how to use the main features of the calculator in the hopes that it will be useful to others.

### 0.1 Initiating the class

To use the calculator, you should download the files `constructor.py` and `EHMMv3.sage` and load them to your Sage file. The file `constructor.py` constructs the class EHMM, while the file `EHMMv3.sage` defines several invariants of hypergeometric series needed for this construction. As such, it is possible to use `EMMv3.sage` without loading `constructor.py`, but the reverse is not possible.

Define an arithmetic hypergeometric datum

$$HD = \{\{a_1, \dots a_n\}, \{b_1, \dots, b_n\}\} = \{\alpha, \beta\}$$

for $a_i, b_i \in \mathbb{Q}$. Note that the length of the $\alpha$ set must equal the length of the $\beta$ set for the calculator to work properly. This differs from many classical references, where the first element of the $\beta$ set is assumed to be 1 and often omitted. For example, the Gaussian hypergeometric function

$${}_2F_1\left[\begin{matrix}1/2 & 1/2 \\ & 1\end{matrix}; x\right]$$

has hypergeometric datum $HD = \{\{1/2, 1/2\}, \{1, 1\}\}$ in our implementation. The datum $HD = \{\{1/2, 1/2\}, \{1\}\}$ corresponds to a different hypergeometric series. In general, the complex-analytic hypergeometric function is defined

$$F(HD, z) = \sum_{k=1}^{\infty} \left( \prod_{i=1}^{n} \frac{(a_i)_k}{(b_i)_k} \right) z^k$$

where $(a)_k = a(1+a)\cdots(a+k-1) = \frac{\Gamma(a+k)}{\Gamma(a)}$, and $\Gamma(\cdot)$ is the Gamma function. Classical hypergeometric functions are defined in Sagemath as

*Please report bugs or direct other questions/concerns about the documentation to Esme Rosen at esmerosen92@gmail.com

`hypergeometric(alpha,beta\{1},z)`

where $\alpha$ is the first multiset in $HD$, and $\beta$ is the second multiset in HD and `beta\{1}` means the set with a 1 removed. In other words, Sagemath uses the ${}_nF_{n+1}(z)$ form of a hypergeometric series, unlike this calculator. Refer to https://doc.sagemath.org/html/en/reference/functions/sage/functions/hypergeometric.html. We establish the following notation. For $a_i, b_j \in \mathbb{Q}$, Let $\alpha^\flat = \{a_1, \cdots, a_n\}, \beta^\flat = \{b_1 = 1, \cdots, b_n\}$, and

$$HD^\flat = \{\alpha^\flat, \beta^\flat\}$$

be a hypergeometric datum. The hypergeometric modularity method then appends a rational number $r$ and $s$ to $\alpha^\flat$ and $\beta^\flat$ respectively to construct the datum

$$HD^\sharp = \{\alpha^\flat \cup \{r\}, \beta^\flat \cup \{s\}\} = \{\alpha^\sharp, \beta^\sharp\}.$$

The modularity aspect requires $F(HD^\flat, t)$ be a modular form for $t$ some modular function.

The EHMM class is given by

`EHMM(multiset1, multiset2, value, hauptmodul, r, s)`

with the input

1. `multiset1` a multiset corresponding to $\alpha^\sharp$.
2. `multiset2` a multiset of the same length as `multiset1` corresponding to $\beta^\sharp$.
3. `value` (*optional*) the rational number $z$ at which the hypergeometric series $F(HD^\sharp, z)$ is evaluated. Since the hypergeometric modularity method is only implemented for $z = 1$, the class assumes $z = 1$. The functionality of the program is limited when $z \neq 1$; namely, `.q_expansion()` defined below will not work.
4. `hauptmodul` (*optional*) any $q$-series. Typically this will by a modular function, corresponding to $t$ as mentioned above. The hauptmoduln for the arithmetic triangle groups listed in [2] are already implemented, as described below. This is allowed to be an arbitrary $q$-series, but the resulting $q$-expansion may not correspond to a modular form if it is not a modular function on the monodromy group for the hypergeometric series.
5. `r` (*optional*) this appends a parameter to `multiset1`, corresponding to $r$ in $HD^\sharp$.
6. `s` (*optional*) this appends a parameter to `multiset2`, corresponding to $s$ in $HD^\sharp$. Inputting `r` and `s` fix `(multiset1,multiset2)` as $HD^\flat$. As such, if `r` and `s` are specified, they **should not be repeated** in `multiset1` and `multiset2`, since the program will append `r` and `s` whenever they are given. For this reason, if you provide one of `r` and `s`, you must provide both.

If `hauptmodul`, `r`, or `s` are not specified, the program will try to guess them based on the tables in the Appendix to [2]. So long as the desired $r$ and $s$ are listed last in `multiset1` and `multiset2` respectively, this typically works well for examples arising from the tables. When working with data or modular functions outside the scope of the tables, these inputs will be required to use the $q$-expansion features.

## 0.2 Simple Commands

Set `HD=EHMM(alpha,beta,z,t,r,s)`.

Here are some basic commands within the family.

1. `HD.alpha` returns `multiset1` with `r` appended if it is specified.
2. `HD.beta` returns `multiset2` with `s` appended if it is specified,

3. `HD.tq(N)` returns $N$ nonzero terms of the $q$-expansion of `hauptmodul` when one is specified or the program is able to identify one in Table 1 of [2]. When $N$ is not specified, it defaults to 10.

```
[8]: HD1=EHMM([1/2,1/2,1/4],[1,1,1])
     HD1.tq(5)

[8]: 16*q - 128*q^2 + 704*q^3 - 3072*q^4 + O(q^5)

[4]: HD1=EHMM([1/2,1/4,1/2],[1,1,1])
     HD1.tq(5)

     Hypergeometric datum does not have corresponding Hauptmodul in Table 1
[4]: ()

[7]: HD1=EHMM([1/2,1/2],[1,1],1,1/4,1)
     HD1.tq(5)

[7]: 16*q - 128*q^2 + 704*q^3 - 3072*q^4 + O(q^5)
```

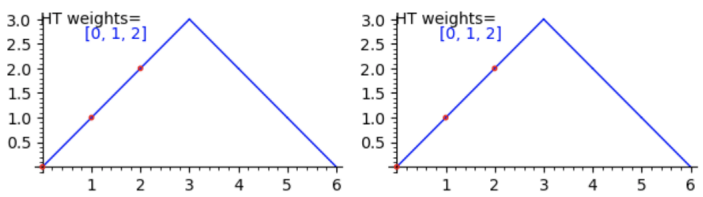
Note that the order of the datum matters when returning the hauptmodul, since the program recognizes the final two elements in each of `multiset1` and `multiset2` as $r$ and $s$. If you are receiving an error, one can always try specifying $r$ and $s$ as demonstrated in the third example. If this does not work, the hauptmodul is not in Table 1, and so must be provided manually.

Many invariants of hypergeometric series are built into the EHMM class. The commands listed below depend only on `alpha` and `beta`, as well as `r` and `s` if they are specified.

1. `HD.denom()` returns the lowest common denominator of of the elements in $\alpha$ and $\beta$, which is usually denoted $M$.
2. `HD.gamma()` returns the $\gamma$ vector of $HD$, which is defined as $\gamma(HD^\flat) = -1 + \sum_{i=1}^{n}(b_i - a_i)$. It is used to determine the local exponents at the singularity 1.
3. `HD.defined_over_Q()` returns `"true"` if the hypergeometric datum is defined over $\mathbb{Q}$ and `"false"` if not. Refer to [11] for a discussion of hypergeometric motives defined over $\mathbb{Q}$.
4. `HD.coeff_p_order(N,p)` for $N$ a positive integer and $p$ a prime returns a graph of the $p$-adic orders of the first $N$ coefficients of the classical hypergeometric series $F(HD, z)$.
5. `HD.zigzag()` returns the zigzag diagram and Hodge-Tate weights for $HD$. Refer to [9] and [4] for the construction. When $z = 0, 1$, the Hodge-Tate weights may be off.
6. `HD.conjugate()` returns a list of hypergeometric data corresponding to the Galois conjugates of the finite field version $\mathbb{P}(HD, z, p)$. See the discussion at the beginning of Section 0.3 for more details.
7. `HD.conjugate_zigzag()` returns zigzag diagrams for all conjugates.

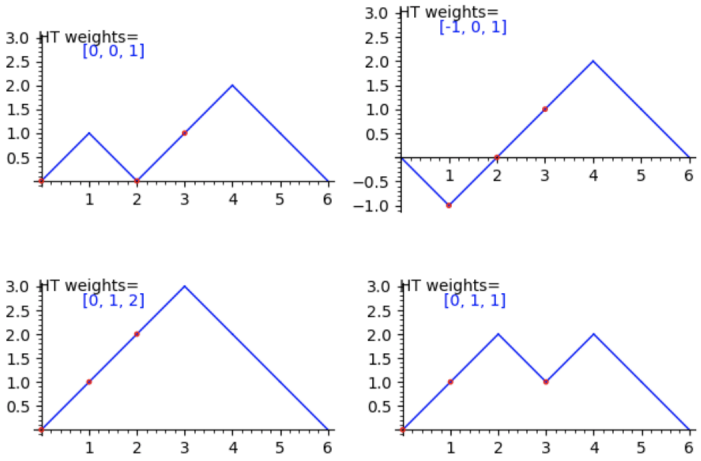
Here is an illustration of the zigzag diagrams for a Galois family.

```
[12]: HD1=EHMM([1/2,1/4,1/2],[1,1,1])
      HD1.conjugate_zigzag()
```



Note that for the attributes in this section, the order of the datum no longer matters. We illustrate with one other example.

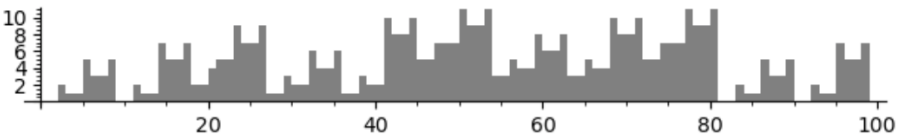
```
[11]: HD1=EHMM([1/2,1/2,1/8],[1,1,3/8])
      HD1.conjugate_zigzag()
```




This provides the zigzag diagrams for a non-Galois family studied in [13]. It is apparent just from looking at the zigzag diagrams that the behavior will be more complicated in this case.

We can also observe the cyclic nature of $p$-adic orders for the coefficients from `HD.coeff_p_order()`.

```
[15]: HD1=EHMM([1/2,1/4,1/2],[1,1,1])
      HD1.coeff_p_order(100,3)
```



## 0.3 Finite Field Hypergeometric Functions

Finite field hypergeometric functions and Jacobi sums are character sums that are the traces of Frobenius for Galois representations which are defined on $G_{\mathbb{Q}(\zeta_M)}$, due to work of Katz [8] and Weil [15] respectively. Therefore, a priori, the input for these functions should be prime ideals $\mathfrak{p}$ in $\mathbb{Q}(\zeta_M)$, and the characters will be characters on the residue field $\mathbb{Z}[\zeta_M]/\mathfrak{p}$. When $\mathfrak{p}$ splits, or equivalently lies above a rational prime $p \equiv 1 \mod M$, the residue field is the finite field $\mathbb{F}_p$. In these cases, the character sum can be implemented using $\mathbb{F}_p$ only, and each choice of generator for $\widehat{\mathbb{F}_p^\times}$ corresponds to a different prime ideal above $p$ in $\mathbb{Z}[\zeta_M]$. In practice, we fix a generator, which is the inverse of the the Teichmüller character, and regard finite field hypergeometric functions at splitting $\mathfrak{p}$ as only depending on a prime $p \equiv 1 \mod M$. We note that if we change the choice of generator, the result will be a Galois conjugate of the original character sum. Thus, the ambiguity introduced by choosing a generator is resolved by the notion of conjugates in the EHMM calculator, which provide the trace value for the various choices of generator by changing the hypergeometric datum instead of the generating character. The conjugates are obtained via `HD.conjugates()`. Note that this is related to but not the same as the concept of conjugate $\mathbb{K}_2(r,s)$ functions introduced in [2], [12], [14]. In this way, the EHMM calculator computes finite field hypergeometric functions for primes $p$ assuming $p \equiv 1 \mod M$ and the Teichmüller character is chosen as a generator. Work of Beukers, Cohen, and Mellit [3] implies Galois

representations attached to hypergeometric data defined over $\mathbb{Q}$ can be extended to $G_{\mathbb{Q}}$, and provides a formula for the values at all primes. This formula is not implemented in the EHMM calculator, since finite field hypergeometric functions for all primes can already be computed in Magma for datum defined over $\mathbb{Q}$. See https://magma.maths.usyd.edu.au/magma/handbook/hypergeometric_motives.

We recall $\widehat{\mathbb{F}_p^\times}$ the set of multiplicative character of a finite field $\mathbb{F}_p$. This group is implemented already in `Sagemath` as `DirichletGroup(p)`. By the discussion above, assuming $p \equiv 1 \mod M$, the character defined by

$$\iota_{r,\mathfrak{p}}(x) \equiv x^{(p-1)\cdot r} \mod \mathfrak{p}$$

for $\mathfrak{p}$ a prime ideal above $p$ in the cyclotomic field $\mathbb{Q}(\zeta_M)$ by Weil [15] can be identified with an element of $\widehat{\mathbb{F}_p^\times}$ when $p$ is fixed. Assume $r$ and $s$ are rational numbers whose least common denominator is $M$. Then we can define the Jacobi sum as

$$J_{\mathfrak{p}}(r,s) = \sum_{x \in \mathbb{F}_p} \iota_{r,\mathfrak{p}}(x) \cdot \iota_{s,\mathfrak{p}}(1-x).$$

This is implemented as

`Jacobi(r,s,p,optional)`

where $r$ and $s$ are rational numbers with least common denominator $M$, and $p$ is a prime congruent to 1 mod $M$. The optional fourth argument can be replaced with `"p_adic"`, which returns the value as a $p$-adic number. An error will be returned if $p \not\equiv 1 \mod M$.

```
[3]: Jacobi(1/4,1/2,5)
[3]: 2*zeta4 + 1
[6]: Jacobi(1/4,1/2,5,"p_adic")
[6]: 3*5 + 4*5^2 + 2*5^3 + 5^4 + 4*5^5 + 2*5^7 + 5^8 + 5^9 + 5^12 + 3*5^13 + 3*5^14 + 5^15 + 4*5^17 + 5^18 + 3*5^19 + O(5^20)
[5]: Jacobi(1/4,1/2,3)
[5]: 'Jacobi sum only implemented for primes equivalent to 1 mod M'
```

Jacobi sums are already implemented in Sagemath as `A.jacobi_sum(B)` when $A$ and $B$ are characters. In particular, our Jacobi sum above is equal to

```
(DirichletGroup(p)[1].bar()^(((p-1)*r).floor())).jacobi_sum(DirichletGroup(r)[1]
    .bar()^(((p-1)*s)).floor()).
```

Greene [6] defined the finite field binomial coefficient as

$$\binom{A}{B} = -B(-1) \cdot J(A, \overline{B}).$$

The binomial coefficient is implemented as

`binomial(r,s,p,optional)`

which returns

$$\binom{\iota_{r,\mathfrak{p}}}{\iota_{s,\mathfrak{p}}}$$

for $r$, $s$, and $p$ as before, and the optional fourth argument again converting the value to a $p$-adic number if `"p_adic"` is entered. This can be used to define a version of the finite field period function, see [5, §4]

$$\mathbb{P}(HD, z, \mathfrak{p}) := \frac{(-1)^n}{q-1} \prod_{i=2}^{n} \iota_{\mathfrak{p},a_i}\iota_{\mathfrak{p},b_i}(-1) \sum_{\chi \in \widehat{k_{\mathfrak{p}}^\times}} \binom{\iota_{\mathfrak{p},a_1}\chi}{\chi} \binom{\iota_{\mathfrak{p},a_2}\chi}{\iota_{\mathfrak{p},b_1}\chi} \ldots \binom{\iota_{\mathfrak{p},a_n}\chi}{\iota_{\mathfrak{p},b_{n-1}}\chi}. \tag{1}$$

The period function is given by

`PP(multiset1,multiset2,value,p,optional)`

where all of the input is as before. A normalized $\mathbb{P}(HD, z, \mathfrak{p})$, denoted as $\mathbb{F}(HD, z, \mathfrak{p})$ in [5], is used by McCarthy [10] and Beukers, Cohen, and Mellit [3]. See [5] for the normalization referred to. This is called $H_p(HD, z)$ or $H(HD, z, \mathfrak{p})$.

Finite field hypergeometric functions can be accessed from the EHMM class as well. They depend on `value`, `r` and `s` if specified, but not on `hauptmodul` .

1. `HD.trace(p,optional)` for $p$ a prime provides the value $\mathbb{P}(HD, \texttt{value}, p)$ as a cyclotomic integer. If you provide the optional second argument of `"p_adic"`, the value will be a $p$-adic integer. Note that this value depends on the order of `multiset1` and `multiset2` If $HD$ is not defined over $\mathbb{Q}$, the function will return an error if $p$ is not congruent to 1 mod $M$. The datum is said to be defined over $\mathbb{Q}$ is the polynomials $\prod_k(x - e^{2a_k\pi i})$ and $\prod_k(x - e^{2b_k\pi i})$ belong to $\mathbb{Z}[x]$. Refer to [3] for more information.

2. `HD.Hp(p,optional)` for $p$ a prime provides the value $H_p(HD, \texttt{value}, p)$ as a cyclotomic number. If you provide the optional second argument of `"p_adic"`, the value will be a $p$-adic number. If $HD$ is not defined over $\mathbb{Q}$, the function will return an error if $p$ is not congruent to 1 mod $M$.

3. `HD.truncated(p,optional)` provides the truncated classical series $F_{p-1}(HD, z)$. Note this truncates at $p - 1$ rather than $p$. If $p$ is a prime, the function provides the value of the truncated series as a $p$-adic integer by default. If $p$ is not a prime, the function the error `'p is not a prime'`. The optional second argument can be replaced with a prime number $q$, and the `HD.truncated(p,q)` produces the series truncated at $p-1$ as a $q$-adic integer. This works regardless of whether $p$ is a prime, so long as $q$ is a prime. The optional second argument can also be replaced with `"rational"` regardless of what $p$ is, and `HD.truncated(p,"rational")` will return $F_{p-1}(HD, z)$ as a rational number.

4. `HD.normalized_Hp(optional)` returns the first few character sums values as 2) of Theorem 2.1 of [2]. For the Galois data and the corresponding Hecke eigenforms listed in Table 1 of [12], Theorem 2.1 of [2] ensures that this will equal the Fourier coefficients of the listed modular form. Changing the argument `optional` will modify the number of primes listed in the table; however, beware that if `optional` is large, the program will run very slowly.

For instance,

[18]:
```
HD1=EHMM([1/2,1/2,1/8],[1,1,1])
print(HD1.trace(73))
print(HD1.Hp(73))
```

```
-58
58
```

.

The function `HD.normalized_Hp()` produces several traces at once, as illustrated below.

```
HD=EHMM([1/2,1/2],[1,1],1/8,1)
HD.normalized_Hp()
```

| p | Normalized Hp |
|---|---|
| 17 | −6 |
| 41 | −66 |
| 73 | −58 |
| 89 | 102 |
| 97 | 26 |
| 113 | 66 |

In general, the values will be cyclotomic integers rather than rational integers. Sometimes, even when $\mathbb{P}(HD, 1, \mathfrak{p})$ is an integer, $H_p(HD, 1, \mathfrak{p})$ is not, as demonstrated below.

```
[7]: HD1=EHMM([1/2,1/2,1/8],[1,1,7/8])
     print(HD1.trace(73))
     print(HD1.Hp(73))
```

```
-10
-60/73*zeta72^15 - 60/73*zeta72^9 + 60/73*zeta72^3 + 10/73
```

In these situation, sometimes it is more useful to convert $H_p$ to a $p$-adic number. For instance, we can convert the example above to be $p$-adic and compare it to `HD.truncated(73)` to demonstrate Theorem 2.1 of [2].

```
[6]: HD1=EHMM([1/2,1/2,1/8],[1,1,7/8])
     print(HD1.Hp(73,"p_adic"))
     print(HD1.truncated(73))
```

```
5 + 56*73 + 9*73^2 + 63*73^3 + 65*73^4 + 10*73^5 + 44*73^6 + 30*73^7 + 60*73^8 + 24*73^9 + 67*73^10 + 32*73^11 + 39*73^12 + 72*73^13 + 14*73^1
4 + 72*73^15 + 4*73^16 + 53*73^17 + 13*73^18 + 47*73^19 + O(73^20)
5 + 56*73 + 41*73^2 + 72*73^3 + 36*73^4 + 7*73^5 + 41*73^6 + 25*73^7 + 54*73^8 + 16*73^9 + 45*73^10 + 43*73^11 + 62*73^13 + 42*73^14 + 4*73^15
+ 2*73^16 + 35*73^17 + 61*73^18 + 7*73^19 + O(73^20)
```

Note that the first two $p$-adic digits are the same. This is what is meant by a mod $p^2$ congruence when the numbers are not integers.

## 0.4 $q$-expansions

The formal series expansion for the classical hypergeometric function can also be accessed via

`Hyp(multiset1, multiset2,value,N)`

The `multiset1` and `multiset2` are the $\alpha$ and $\beta$ sets for the hypergeometric datum. Since this function is set up to be a power series, the parameter $z$ in `value` must be a power series or a polynomial. The value $F_N(HD; z_0)$ for $z_0$ a constant should be obtained using the command

`EHMM(HD;z_0).truncated(N+1)`

As noted earlier, $F(HD^\flat, z)$ is a modular form when $HD^\flat$ corresponds to an entry in Table 1 of [2] and $z$ is replaced with an appropriate hauptmodul. For example, if $\lambda$ is the modular lambda function,

$$F(\{1/2, 1/2\}, \{1, 1\}; \lambda(q)) = \theta_3^2(\tau) = \sum_{n_1, n_2 \in \mathbb{Z}} q^{n_1^2 + n_2^2} \qquad \text{where } q = e^{2\pi i \tau},$$

is a weight-1 modular form.

```
### Loding packages
load('constructor.py');load('EHMMv3.sage')
Hyp([1/2,1/2],[1,1],q,5)
```

```
1 + 1/4*q + 9/64*q^2 + 25/256*q^3 + 1225/16384*q^4
```

```
t2.0(5) #call modular lambda function, denoted by t2
```

```
16*q - 128*q^2 + 704*q^3 - 3072*q^4 + O(q^5)
```

```
Hyp([1/2,1/2],[1,1],t2,10).0(10)
```

```
1 + 4*q + 4*q^2 + 4*q^4 + 8*q^5 + 4*q^8 + 4*q^9 + O(q^10)
```

```
theta3=sum(q^(n^2) for n in srange(-10,10)) # define theta3
(theta3^2).0(10)
```

```
1 + 4*q + 4*q^2 + 4*q^4 + 8*q^5 + 4*q^8 + 4*q^9 + O(q^10)
```

The integral formula says that

$$\frac{1}{B(r,s-r)}\int_0^1 z^{r-1}(1-z)^{s-r-1}F(HD^\flat,z)dz = F(HD^\sharp,1).$$

The modular form constructed by the hypergeometric modularity method is the integrand up to a constant, $t^{r-1}(1-t)^{s-r-1}F(HD^\flat,t)dt$, where $t$ becomes the appropriate hauptmodul. To compute $q$-expansions of these modular forms, all of the input data for EHMM class is required. In particular, the logarithmic derivative of $t(q)$ is a weight-2 modular form if $t(q)$ is a Hauptmodul.

```
(q*diff(t2,q)/t2).O(8) #logarithmic derivative of modular lambda t2
```

```
1 - 8*q + 24*q^2 - 32*q^3 + 24*q^4 - 48*q^5 + 96*q^6 - 64*q^7 + O(q^8)
```

We can verify that this function is actually $\theta_3^4(\tau)$ as well.

```
[10]: Hyp([1/2,1/2],[1,1],t2,8)^2
```

```
[10]: 1 + 8*q + 24*q^2 + 32*q^3 + 24*q^4 + 48*q^5 + 96*q^6 + 64*q^7 + O(q^8)
```

As noted above, sometimes `r`, `s`, and `hauptmodul` can be omitted when they are implemented in Table 1, but these inputs are still essential in the background.

1. `HD.q_expansion(N)` returns the $q$-expansion constructed via the EHMM with $N$ nonzero terms. Note this may not be the $q$-expansion of a congruence or holomorphic modular form. If $N$ is omitted, the command assumes $N = 6$. The value for $N$ currently has a maximum value of 100. It is now possible to specify any $q$-series $Q$ with rational coefficients for `hauptmodul` and return the formal power series

$$Q^{1-r}(1-Q)^{s-r-1}F(HD^\flat,Q)\frac{dQ}{dq}$$

normalized so the first coefficient is 1. If $Q$ is not a hauptmodul on the monodromy group of the underlying ${}_2F_1(z)$ series, as is the case for the data listed in Table 1 of [2], this will not be a modular form. This feature is intended purely for experimental purposes.

2. `HD.conjugate_exp(N)` returns the $q$-expansions for all of the conjugates given via `HD.conjugate()` at once each with $N$ nonzero terms. If $N$ is omitted, the command assumes $N = 6$. The value for $N$ currently has a maximum value of 100. Extra care is required when using this command if the family is non-Galois, since the Galois conjugates do not have a one-to-one correspondence with holomorphic differentials as in the Galois families. In these cases, some $q$-series returned will be not be holomorphic, while other holomorphic conjugates will not appear. Refer to [13] for details.

```
#either define the class using HD^\flat,r,s
HD=EHMM([1/2,1/2],[1,1],1/8,1) # or equivalently HD=EHMM([1/2,1/2,1/8],[1,1,1])
print(HD.alpha,HD.beta,HD.denominator())
print(HD.tq().O(7)) #the default Hauptmodul
print(table(HD.conjugate())) #print conjugates in a table
```

```
[1/2, 1/2, 1/8] [1, 1, 1] 8
16*q - 128*q^2 + 704*q^3 - 3072*q^4 + 11488*q^5 - 38400*q^6 + O(q^7)
  [1/2, 1/2, 1/8]   [1, 1, 1]
  [1/2, 1/2, 3/8]   [1, 1, 1]
  [1/2, 1/2, 5/8]   [1, 1, 1]
  [1/2, 1/2, 7/8]   [1, 1, 1]
```

```
HD.conjugate_exp() #print first few q-terms of the conjugate K2(r,s)
```

```
[q - 3*q^9 - 6*q^17 + 23*q^25 + 12*q^33 + O(q^40),
 q^3 - q^11 - 7*q^19 + 6*q^27 + 16*q^35 + O(q^40),
 q^5 + q^13 - 4*q^21 - 3*q^29 + q^37 + O(q^40),
 q^7 + 3*q^15 + 3*q^23 + 4*q^31 + 3*q^39 + O(q^40)]
```

## 0.5 Other built in $q$-series

Several other $q$-series are built in for the purposes of demonstrations. We record these here.

1. The function `eta(n,N)` for $n$ and $N$ both positive integers returns the expanded form of

$$\prod_{k=1}^{N}(1-q^{nk}).$$

   This is useful for building other eta quotients, though note the term $q^{1/24}$ appearing in the eta function is omitted.

2. The function `K2(r,s, optional)` returns the function $\mathbb{K}_2(r,s)(N\tau)$ introduced in [2] and classified in [12]. The optional third argument specifies the number of Fourier coefficients.

3. The function `K1(r,s, optional)` returns the function $\mathbb{K}_1(r,s)(N\tau)$ introduced in [14]. The optional third argument specifies the number of Fourier coefficients.

4. The function `K3(r,s, optional)` returns the function $\mathbb{K}_3(r,s)(N\tau)$ introduced in [2] and classified in [7]. The optional third argument specifies the number of Fourier coefficients.

The hauptmoduln in Table 1 of [2] are given the following names in the calculator, for reference.

| Group | Name in [2] | Code |
|---|---|---|
| $\Gamma_0(1)$ | $1728/j$ | `Jinv` |
| $\Gamma_0(2)$ | $u$ | `uu` |
| $\Gamma_0(3)$ | $t_3$ | `t3` |
| $\Gamma_0(4)$ | $t_2$ | `t2` |
| $\Gamma_0(2)^+$ | $t_{4+}$ | `t4p` |
| $\Gamma_0(3)^+$ | $t_{6+}$ | `t6p` |

Table 1: Code for hauptmoduln in EHMM calculator

```
show(Jinv.0(5))
show(uu.0(5))
show(t2.0(5))
show(t3.0(5))
show(t4p.0(5))
show(t6p.0(5))
```

$1728q - 1285632q^2 + 616294656q^3 - 242544070656q^4 + O(q^5)$
$-64q - 1536q^2 - 19200q^3 - 167936q^4 + O(q^5)$
$16q - 128q^2 + 704q^3 - 3072q^4 + O(q^5)$
$27q - 405q^2 + 4617q^3 - 45333q^4 + O(q^5)$
$256q - 26624q^2 + 1649664q^3 - 79806464q^4 + O(q^5)$
$108q - 4536q^2 + 105948q^3 - 1834704q^4 + O(q^5)$

# References


[1] Michael Allen, Brian Grove, Ling Long, and Fang-Ting Tu. The Explicit Hypergeometric-Modularity Method II, 2024 arXiv: 2411.15116.

[2] Michael Allen, Brian Grove, Ling Long, and Fang-Ting Tu. The explicit hypergeometric-modularity method I. *Advances in Mathematics*, 478:110411, 2025.

[3] Frits Beukers, Henri Cohen, and Anton Mellit. Finite hypergeometric functions. *Pure Appl. Math. Q.*, 11(4):559–589, 2015.

[4] Roman Fedorov. Variations of Hodge structures for hypergeometric differential operators and parabolic Higgs bundles. *Int. Math. Res. Not. IMRN*, (18):5583–5608, 2018.

[5] Jenny Fuselier, Ling Long, Ravi Ramakrishna, Holly Swisher, and Fang-Ting Tu. Hypergeometric functions over finite fields. *Mem. Amer. Math. Soc.*, 280(1382), 2022.

[6] John Greene. Hypergeometric functions over finite fields. *Trans. Amer. Math. Soc.*, 301(1):77–101, 1987.

[7] Brian Grove. On Some Hypergeometric Modularity Conjectures of Dawsey and McCarthy, 2025, arXiv: 2507.19971.

[8] Nicholas M. Katz. *Exponential sums and differential equations*, volume 124 of *Annals of Mathematics Studies*. Princeton University Press, Princeton, NJ, 1990.

[9] Ling Long and Yifan Yang. Hodge numbers of hypergeometric data, 2024, arxiv:2404.02834.

[10] Dermot McCarthy. Transformations of well-poised hypergeometric functions over finite fields. *Finite Fields Appl.*, 18(6):1133–1147, 2012.

[11] David P. Roberts and Fernando Rodriguez Villegas. Hypergeometric motives. *Notices Amer. Math. Soc.*, 69(6):914–929, 2022.

[12] Esme Rosen. $L$-values of certain weight 3 Modular Forms and Transformations of Hypergeometric Series, 2024, arxiv:2412.07054.

[13] Esme Rosen. 'Mixed' CM structures associated to certain Hypergeometric Representations. *In Preparation*, 2025.

[14] Esme Rosen. Modular forms and certain ${}_2F_1(1)$ hypergeometric series, 2025, arXiv:2502.08760.

[15] André Weil. Jacobi sums as "Grössencharaktere". *Trans. Amer. Math. Soc.*, 73:487–495, 1952.