Anderson Motives

Release 10.10

The Sage Development Team

Oct 28, 2026

CONTENTS

1 Anderson motives 1

2 Indices and Tables 15

Python Module Index 17

Index 19

CHAPTER

ONE

ANDERSON MOTIVES

Let $\mathbf{F}_q[T]$ be a polynomial ring with coefficients in a finite field $\mathbf{F}_q$ and let K be an extension of $\mathbf{F}_q$ equipped with a distinguished element z.

By definition, an Anderson motive attached to these data is a free module of finite rank M over $K[T]$, equipped with a linear automorphism

$$\tau_M : \tau^\star M \left[\frac{1}{T-z}\right] \to M \left[\frac{1}{T-z}\right]$$

where $\tau^\star M = K \otimes_{K,\mathrm{Frob}} M$ and the notation means that K is viewed as an algebra over itself through the Frobenius $\mathrm{Frob} : x \mapsto x^q$.

Anderson motives attached to Drinfeld modules

Any Drinfeld module ϕ over (A, γ) with $\gamma : A \to K, T \mapsto z$ gives rise to an Anderson motive. By definition, it is $M(\phi) := K\{\tau\}$ (the ring of Ore polynomials with commutation rule $\tau\lambda = \lambda^q\tau$ for $\lambda \in K$) where

- the structure of $\mathbf{F}_q[T]$-module is given by right multiplication by ϕ_a ($a \in \mathbf{F}_q[T]$),
- the structure of K-module is given by left multiplication,
- the automorphism $\tau_{M(\phi)}$ is the left multiplication by τ in the Ore polynomial ring.

Anderson motives are nevertheless much more general than Drinfeld modules. Besides, their linear nature allows for importing many interesting construction of linear and bilinear algebra.

In SageMath, one can create the Anderson motive corresponding to a Drinfeld module as follows:

```
sage: k = GF(5)
sage: A.<T> = k[]
sage: K.<z> = k.extension(3)
sage: phi = DrinfeldModule(A, [z, z^2, z^3, z^4])
sage: M = phi.anderson_motive()
sage: M
Anderson motive of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^
↪2 + z^2*τ + z
```

```
>>> from sage.all import *
>>> k = GF(Integer(5))
>>> A = k['T']; (T,) = A._first_ngens(1)
>>> K = k.extension(Integer(3), names=('z',)); (z,) = K._first_ngens(1)
>>> phi = DrinfeldModule(A, [z, z**Integer(2), z**Integer(3), z**Integer(4)])
>>> M = phi.anderson_motive()
>>> M
```

(continues on next page)

(continued from previous page)

```
Anderson motive of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^
↪2 + z^2*τ + z
```

We see that M has rank 3; it is actually a general fact that the Anderson motive attached a Drinfeld module has the same rank than the underlying Drinfeld module.

The canonical basis corresponds to the vectors τ^i for i varying between 0 and $r-1$ where r is the rank:

```
sage: tau = phi.ore_variable()
sage: M(tau^0)
(1, 0, 0)
sage: M(tau^1)
(0, 1, 0)
sage: M(tau^2)
(0, 0, 1)
```

```
>>> from sage.all import *
>>> tau = phi.ore_variable()
>>> M(tau**Integer(0))
(1, 0, 0)
>>> M(tau**Integer(1))
(0, 1, 0)
>>> M(tau**Integer(2))
(0, 0, 1)
```

Note

We warn the use that the syntax $M(1)$ fails because the argument is not in $K\{\tau\}$, and no automatic coercion is performed here.

Higher powers of τ can be rewritten as linear combinations (over $K[T]$!) of those three ones:

```
sage: M(tau^3)
((z^2 + 3*z)*T + 2*z^2 + 3*z + 3, 3*z^2 + 2*z + 4, 2*z^2 + 1)
sage: M(tau^4)
((4*z^2 + 4*z + 3)*T + z^2 + 4*z + 2, (z^2 + 4*z)*T + 3, 3*z^2 + 4*z + 4)
```

```
>>> from sage.all import *
>>> M(tau**Integer(3))
((z^2 + 3*z)*T + 2*z^2 + 3*z + 3, 3*z^2 + 2*z + 4, 2*z^2 + 1)
>>> M(tau**Integer(4))
((4*z^2 + 4*z + 3)*T + z^2 + 4*z + 2, (z^2 + 4*z)*T + 3, 3*z^2 + 4*z + 4)
```

The matrix of the operator τ_M can be obtained using the method `matrix()`:

```
sage: M.matrix()
[                                0                                1                ⌴
↪          0]
[                                0                                0                ⌴
↪          1]
```

(continues on next page)

(continued from previous page)

```
[(z^2 + 3*z)*T + 2*z^2 + 3*z + 3                    3*z^2 + 2*z + 4                    ␣
↪ 2*z^2 + 1]
```

```
>>> from sage.all import *
>>> M.matrix()
[                                 0                                 1                 ␣
↪          0]
[                                 0                                 0                 ␣
↪          1]
[(z^2 + 3*z)*T + 2*z^2 + 3*z + 3                    3*z^2 + 2*z + 4                    ␣
↪ 2*z^2 + 1]
```

Note

Here, as it is conventional in SageMath, we use the row representation, meaning that the coordinates of the image by $\tau_M(\tau^i)$ are written in the i-th row.

SageMath provides facilities to pick elements in M and perform basic operations with them:

```
sage: u, v, w = M.basis()
sage: T*u + z*w
(T, 0, z)
sage: w.image()  # image by tau_M
((z^2 + 3*z)*T + 2*z^2 + 3*z + 3, 3*z^2 + 2*z + 4, 2*z^2 + 1)
```

```
>>> from sage.all import *
>>> u, v, w = M.basis()
>>> T*u + z*w
(T, 0, z)
>>> w.image()  # image by tau_M
((z^2 + 3*z)*T + 2*z^2 + 3*z + 3, 3*z^2 + 2*z + 4, 2*z^2 + 1)
```

It is also possible to give names to the vectors of the canonical basis and then use when printing:

```
sage: psi = DrinfeldModule(A, [z, z+1, z+2])
sage: N.<e0, e1> = psi.anderson_motive()
sage: N.random_element()  # random
((4*z+4)*T^2+(3*z^2+1)*T+z^2+3*z+3)*e0 + (T^2+(2*z^2+1)*T+3*z^2)*e1
```

```
>>> from sage.all import *
>>> psi = DrinfeldModule(A, [z, z+Integer(1), z+Integer(2)])
>>> N = psi.anderson_motive(names=('e0', 'e1',)); (e0, e1,) = N._first_ngens(2)
>>> N.random_element()  # random
((4*z+4)*T^2+(3*z^2+1)*T+z^2+3*z+3)*e0 + (T^2+(2*z^2+1)*T+3*z^2)*e1
```

More Anderson motives

One can also build the dual of the Anderson motive attached to a Drinfeld simply by setting the attribute `dual=True`:

```
sage: Md = phi.anderson_motive(dual=True)
sage: Md
Dual Anderson motive of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z +␣
↪2)*τ^2 + z^2*τ + z
sage: Md.matrix()
[           z^2/(T + 4*z)                          1                          0]
[     (2*z + 2)/(T + 4*z)                          0                          1]
[(2*z^2 + 2*z)/(T + 4*z)                          0                          0]
```

```
>>> from sage.all import *
>>> Md = phi.anderson_motive(dual=True)
>>> Md
Dual Anderson motive of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z +␣
↪2)*τ^2 + z^2*τ + z
>>> Md.matrix()
[           z^2/(T + 4*z)                          1                          0]
[     (2*z + 2)/(T + 4*z)                          0                          1]
[(2*z^2 + 2*z)/(T + 4*z)                          0                          0]
```

We observe that some entries of the previous matrix have denominator $T - z$. This corresponds to the fact that τ_M is only defined after inverting $T - z$ in full generality, and it implies in particular that `Md` does not come itself from a Drinfeld module.

Finally, SageMath also provides a general constructor *`AndersonMotive()`* which allows in particular to explicitly provide the matrix of τ_M:

```
sage: mat = matrix(2, 2, [[T, z], [1, 1]])
sage: N = AndersonMotive(A, mat)
sage: N
Anderson motive of rank 2 over Univariate Polynomial Ring in T over Finite Field in z␣
↪of size 5^3
sage: N.matrix()
[T z]
[1 1]
```

```
>>> from sage.all import *
>>> mat = matrix(Integer(2), Integer(2), [[T, z], [Integer(1), Integer(1)]])
>>> N = AndersonMotive(A, mat)
>>> N
Anderson motive of rank 2 over Univariate Polynomial Ring in T over Finite Field in z␣
↪of size 5^3
>>> N.matrix()
[T z]
[1 1]
```

Getters to context objects

There are many rings and ring extensions associated to an Anderson motive. For the convenience of the reader, we hereby list the methods that can be used to retrieve them. We stress that some of the methods below share their name with methods of the class `DrinfeldModule` while returning different objects (see Rubric *Getters to context objects*). This is because contrary to Drinfeld modules, Anderson motives are implemented as modules (more precisely, as Ore modules).

- `base()`: the polynomial ring $K[T]$; this is because we implement Anderson motives as modules over the ring $K[T]$
- `base_ring()`: a reference to `base()`.
- `base_field()`: the fraction field of the `base()`, *i.e.* $K(T)$
- `function_ring()`: see Rubric *Getters to context objects* in class `DrinfeldModule`.
- `A_field()`: see Rubric *Getters to context objects* in class `DrinfeldModule`.
- `ore_polring()`: the Ore polynomial ring $K[T]\{\tau\}$; in particular, the `ore_polring()` for the class *`AndersonMotive`* is strictly larger than the `ore_polring()` for the class `DrinfeldModule`.
- `ore_ring()`: the same Ore polynomial ring, but in the variable x; this method is inherited from the class `sage.modules.ore_module.OreModule`.

Morphisms between Anderson motives

By definition, a morphism between Anderson motives is a $A \otimes K$-linear morphism commuting with the action of τ.

One important class of morphisms of Anderson motives are those coming from isogenies between Drinfeld modules. Such morphisms can be built easily as follows:

```
sage: u = phi.hom(tau + z)
sage: u
Drinfeld Module morphism:
  From: Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^2 + z^2*τ + z
  To:   Drinfeld module defined by T |--> (4*z^2 + 2*z + 4)*τ^3 + (4*z^2 + 1)*τ^2 + (z^2 + 2)*τ + z
  Defn: τ + z
sage: Mu = u.anderson_motive()
sage: Mu
Morphism:
  From: Anderson motive of Drinfeld module defined by T |--> (4*z^2 + 2*z + 4)*τ^3 + (4*z^2 + 1)*τ^2 + (z^2 + 2)*τ + z
  To:   Anderson motive of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^2 + z^2*τ + z
sage: Mu.matrix()
[                              z                                 1                  0]
[                              0                 2*z^2 + 4*z + 4                  1]
[(z^2 + 3*z)*T + 2*z^2 + 3*z + 3                 3*z^2 + 2*z + 4                  2]
```

```
>>> from sage.all import *
>>> u = phi.hom(tau + z)
>>> u
Drinfeld Module morphism:
  From: Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^2 + z^2*τ + z
  To:   Drinfeld module defined by T |--> (4*z^2 + 2*z + 4)*τ^3 + (4*z^2 + 1)*τ^2 + (z^2 + 2)*τ + z
  Defn: τ + z
>>> Mu = u.anderson_motive()
```

(continues on next page)

(continued from previous page)

```
>>> Mu
Morphism:
  From: Anderson motive of Drinfeld module defined by T |--> (4*z^2 + 2*z + 4)*τ^3 +␣
↪(4*z^2 + 1)*τ^2 + (z^2 + 2)*τ + z
  To:   Anderson motive of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z␣
↪+ 2)*τ^2 + z^2*τ + z
>>> Mu.matrix()
[                               z                                   1                  ␣
↪          0]
[                               0                 2*z^2 + 4*z + 4                      ␣
↪          1]
[(z^2 + 3*z)*T + 2*z^2 + 3*z + 3                 3*z^2 + 2*z + 4                      ␣
↪          2]
```

Standard methods of linear algebra are available:

```
sage: Mu.is_injective()
True
sage: Mu.is_surjective()
False
sage: Mu.image().basis()
[(T + 3, 0, 0), (z, 1, 0), (z^2 + 2*z + 1, 0, 1)]
```

```
>>> from sage.all import *
>>> Mu.is_injective()
True
>>> Mu.is_surjective()
False
>>> Mu.image().basis()
[(T + 3, 0, 0), (z, 1, 0), (z^2 + 2*z + 1, 0, 1)]
```

We check below that the characteristic polynomial of the Frobenius of ϕ is equal to the characteristic polynomial of the action of the Frobenius on the motive:

```
sage: f = phi.frobenius_endomorphism()
sage: f
Endomorphism of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^2 +␣
↪z^2*τ + z
  Defn: τ^3
sage: Mf = f.anderson_motive()
sage: Mf.characteristic_polynomial()
X^3 + (T + 4)*X^2 + 3*T^2*X + 4*T^3 + 2*T + 2
```

```
>>> from sage.all import *
>>> f = phi.frobenius_endomorphism()
>>> f
Endomorphism of Drinfeld module defined by T |--> (2*z^2 + 2*z)*τ^3 + (2*z + 2)*τ^2 +␣
↪z^2*τ + z
  Defn: τ^3
>>> Mf = f.anderson_motive()
>>> Mf.characteristic_polynomial()
X^3 + (T + 4)*X^2 + 3*T^2*X + 4*T^3 + 2*T + 2
```

```
sage: phi.frobenius_charpoly()
X^3 + (T + 4)*X^2 + 3*T^2*X + 4*T^3 + 2*T + 2
```

```
>>> from sage.all import *
>>> phi.frobenius_charpoly()
X^3 + (T + 4)*X^2 + 3*T^2*X + 4*T^3 + 2*T + 2
```

AUTHOR:

- Xavier Caruso, Antoine Leudière (2025-11): initial version

sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonMotive**(*arg1*, *arg2=None*, *names=None*)

Construct an Anderson motive.

INPUT:

- `arg1`, `arg2` – arguments defining the Anderson motive, they can be:
  - a Drinfeld module and `None`
  - the underlying function ring A (which currently needs to be of the form $\mathbf{F}_q[t]$) and a A-field K; these parameters correspond to the trivial Anderson motive over $A \otimes K$
  - the underlying function ring A and an element z in it; the A-field is then the parent K of z viewed as an algebra over A through $A \mapsto K, T \mapsto z$, and the returned Anderson motive is again the trivial one over $A \otimes K$
  - A and τ where:
    - A is either $\mathbf{F}_q[t]$ or a category (of Drinfeld modules or Anderson motives)
    - τ is the matrix defining the Anderson motive
- `names` – a string or a list of strings (default: `None`), the names of the vectors of the canonical basis; if `None`, elements will be represented as row vectors

EXAMPLES:

```
sage: A.<T> = GF(7)[]
sage: K.<z> = GF(7^3)
```

```
>>> from sage.all import *
>>> A = GF(Integer(7))['T']; (T,) = A._first_ngens(1)
>>> K = GF(Integer(7)**Integer(3), names=('z',)); (z,) = K._first_ngens(1)
```

We first construct the trivial Anderson motive over K:

```
sage: M = AndersonMotive(A, K)
sage: M
Anderson motive of rank 1 over Univariate Polynomial Ring in T over Finite Field␣
↪in z of size 7^3
sage: M.matrix()
[1]
```

```
>>> from sage.all import *
>>> M = AndersonMotive(A, K)
>>> M
```

(continues on next page)

(continued from previous page)

```
Anderson motive of rank 1 over Univariate Polynomial Ring in T over Finite Field␣
↪in z of size 7^3
>>> M.matrix()
[1]
```

Here the structure of A-field on K is given by the map that takes T to the canonical generator of K, namely z:

```
sage: M.A_field()
Finite Field in z of size 7^3 over its base
sage: M.A_field().defining_morphism()
Ring morphism:
  From: Univariate Polynomial Ring in T over Finite Field of size 7
  To:   Finite Field in z of size 7^3 over its base
  Defn: T |--> z
```

```
>>> from sage.all import *
>>> M.A_field()
Finite Field in z of size 7^3 over its base
>>> M.A_field().defining_morphism()
Ring morphism:
  From: Univariate Polynomial Ring in T over Finite Field of size 7
  To:   Finite Field in z of size 7^3 over its base
  Defn: T |--> z
```

Specifying another element in K leads to a different structure of A-field:

```
sage: N = AndersonMotive(A, z^2)
sage: N.A_field().defining_morphism()
Ring morphism:
  From: Univariate Polynomial Ring in T over Finite Field of size 7
  To:   Finite Field in z of size 7^3 over its base
  Defn: T |--> z^2
```

```
>>> from sage.all import *
>>> N = AndersonMotive(A, z**Integer(2))
>>> N.A_field().defining_morphism()
Ring morphism:
  From: Univariate Polynomial Ring in T over Finite Field of size 7
  To:   Finite Field in z of size 7^3 over its base
  Defn: T |--> z^2
```

One can also directly construct the Anderson motive attached to a Drinfeld module as follows:

```
sage: phi = DrinfeldModule(A, [z, z^2, z^3])
sage: AndersonMotive(phi)
Anderson motive of Drinfeld module defined by T |--> (z^2 + 3)*τ^2 + z^2*τ + z
```

```
>>> from sage.all import *
>>> phi = DrinfeldModule(A, [z, z**Integer(2), z**Integer(3)])
>>> AndersonMotive(phi)
Anderson motive of Drinfeld module defined by T |--> (z^2 + 3)*τ^2 + z^2*τ + z
```

Finally, another possibility is to give the matrix of τ as an argument:

```
sage: tau = matrix(2, 2, [[T, z], [z+1, 1]])
sage: tau
[    T     z]
[z + 1     1]
sage: M = AndersonMotive(A, tau)
sage: M
Anderson motive of rank 2 over Univariate Polynomial Ring in T over Finite Field␣
↪in z of size 7^3
sage: M.matrix()
[    T     z]
[z + 1     1]
```

```
>>> from sage.all import *
>>> tau = matrix(Integer(2), Integer(2), [[T, z], [z+Integer(1), Integer(1)]])
>>> tau
[    T     z]
[z + 1     1]
>>> M = AndersonMotive(A, tau)
>>> M
Anderson motive of rank 2 over Univariate Polynomial Ring in T over Finite Field␣
↪in z of size 7^3
>>> M.matrix()
[    T     z]
[z + 1     1]
```

In this case, the structure of A-field is automatically inferred:

```
sage: M.A_field().defining_morphism()
Ring morphism:
  From: Univariate Polynomial Ring in T over Finite Field of size 7
  To:   Finite Field in z of size 7^3 over its base
  Defn: T |--> z^2 + z
```

```
>>> from sage.all import *
>>> M.A_field().defining_morphism()
Ring morphism:
  From: Univariate Polynomial Ring in T over Finite Field of size 7
  To:   Finite Field in z of size 7^3 over its base
  Defn: T |--> z^2 + z
```

See also

sage.rings.function_field.drinfeld_modules.anderson_motive

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonMotiveMorphism**(*parent*, *im_gens*, *check=True*)

Bases: OreModuleMorphism

A class for morphisms betweeen Anderson motives.

characteristic_polynomial(*var='X'*)

Return the characteristic polynomial of this morphism.

INPUT:

- `var` – a string (default: `X`), the name of the variable

EXAMPLES:

```
sage: A.<T> = GF(5)[]
sage: K.<z> = GF(5^3)
sage: phi = DrinfeldModule(A, [z, z^2, z^3])
sage: f = phi.scalar_multiplication(T).anderson_motive()
sage: chi = f.characteristic_polynomial()
sage: chi
X^2 + 3*T*X + T^2
sage: chi.factor()
(4*X + T)^2
```

```
>>> from sage.all import *
>>> A = GF(Integer(5))['T']; (T,) = A._first_ngens(1)
>>> K = GF(Integer(5)**Integer(3), names=('z',)); (z,) = K._first_ngens(1)
>>> phi = DrinfeldModule(A, [z, z**Integer(2), z**Integer(3)])
>>> f = phi.scalar_multiplication(T).anderson_motive()
>>> chi = f.characteristic_polynomial()
>>> chi
X^2 + 3*T*X + T^2
>>> chi.factor()
(4*X + T)^2
```

We compute the characteristic polynomial of the Frobenius and compare the result with the output of the method `sage.rings.function_field.drinfeld_modules.drinfeld_module_finite.DrinfeldModule_finite.frobenius_charpoly()`:

```
sage: Frob = phi.frobenius_endomorphism().anderson_motive()
sage: Frob.characteristic_polynomial()
X^2 + X + 3*T^3 + 4*T + 4
sage: phi.frobenius_charpoly()
X^2 + X + 3*T^3 + 4*T + 4
```

```
>>> from sage.all import *
>>> Frob = phi.frobenius_endomorphism().anderson_motive()
>>> Frob.characteristic_polynomial()
X^2 + X + 3*T^3 + 4*T + 4
>>> phi.frobenius_charpoly()
X^2 + X + 3*T^3 + 4*T + 4
```

charpoly(*var='X'*)

alias of *characteristic_polynomial()*.

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonMotive_drinfeld**(*mat*, *ore*, *denominator*, *names*, *category*, *phi*, *dual*)

Bases: *AndersonMotive_general*

A class for Anderson motives coming from Drinfeld modules.

drinfeld_module()

Return the Drinfeld module from which this Anderson motive was constructed.

EXAMPLES:

```
sage: A.<T> = GF(5)[]
sage: K.<z> = GF(5^5)
sage: phi = DrinfeldModule(A, [z, z^2, z^3])
sage: M = phi.anderson_motive()
sage: M.drinfeld_module()
Drinfeld module defined by T |--> z^3*τ^2 + z^2*τ + z
sage: M.drinfeld_module() is phi
True
```

```
>>> from sage.all import *
>>> A = GF(Integer(5))['T']; (T,) = A._first_ngens(1)
>>> K = GF(Integer(5)**Integer(5), names=('z',)); (z,) = K._first_ngens(1)
>>> phi = DrinfeldModule(A, [z, z**Integer(2), z**Integer(3)])
>>> M = phi.anderson_motive()
>>> M.drinfeld_module()
Drinfeld module defined by T |--> z^3*τ^2 + z^2*τ + z
>>> M.drinfeld_module() is phi
True
```

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonMotive_general**(*mat*, *ore*, *denominator*, *names*, *category*)

Bases: OreModule

General class for Anderson motives.

hodge_pink_weights()

Return the Hodge-Pink weights of this Anderson motive, sorted by increasing order.

EXAMPLES:

```
sage: A.<T> = GF(5)[]
sage: K.<z> = GF(5^4)
sage: phi = DrinfeldModule(A, [z, z^2, z^3, z^4])
sage: M = phi.anderson_motive()
sage: M.hodge_pink_weights()
[0, 0, 1]
```

```
>>> from sage.all import *
>>> A = GF(Integer(5))['T']; (T,) = A._first_ngens(1)
>>> K = GF(Integer(5)**Integer(4), names=('z',)); (z,) = K._first_ngens(1)
>>> phi = DrinfeldModule(A, [z, z**Integer(2), z**Integer(3), z**Integer(4)])
>>> M = phi.anderson_motive()
>>> M.hodge_pink_weights()
[0, 0, 1]
```

We check that the Hodge-Pink weights of the dual are the opposite of the Hodge-Pink weights of the initial Anderson motive:

```
sage: N = phi.anderson_motive(dual=True)
sage: N.hodge_pink_weights()
[-1, 0, 0]
```

```
>>> from sage.all import *
>>> N = phi.anderson_motive(dual=True)
>>> N.hodge_pink_weights()
[-1, 0, 0]
```

REFERENCES:

For definition and relevance of this notion, we refer to [HJ2020].

is_effective()

Return whether this Anderson module is effective, that is, whether the action of τ stabilizes it. This is also equivalent to the fact that all Hodge-Pink weights are nonnegative.

EXAMPLES:

```
sage: A.<T> = GF(5)[]
sage: K.<z> = GF(5^4)
sage: phi = DrinfeldModule(A, [z, z^2, z^3])
sage: M = phi.anderson_motive()
sage: M.is_effective()
True
```

```
>>> from sage.all import *
>>> A = GF(Integer(5))['T']; (T,) = A._first_ngens(1)
>>> K = GF(Integer(5)**Integer(4), names=('z',)); (z,) = K._first_ngens(1)
>>> phi = DrinfeldModule(A, [z, z**Integer(2), z**Integer(3)])
>>> M = phi.anderson_motive()
```

(continues on next page)

(continued from previous page)

```
>>> M.is_effective()
True
```

```
sage: N = phi.anderson_motive(dual=True)
sage: N.is_effective()
False
```

```
>>> from sage.all import *
>>> N = phi.anderson_motive(dual=True)
>>> N.is_effective()
False
```

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonMotive_homspace**(*domain*, *codomain*, *category=None*)

Bases: OreModule_homspace

Element

alias of *AndersonMotiveMorphism*

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonQuotientMotive**(*cover*, *submodule*, *names*)

Bases: *AndersonMotive_general*, OreQuotientModule

A class for Anderson motives defined as quotients of an other Anderson motive.

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonSubMotive**(*ambient*, *submodule*, *names*)

Bases: *AndersonMotive_general*, OreSubmodule

A class for Anderson motives defined as submodules of an other Anderson motive.

class sage.rings.function_field.drinfeld_modules.anderson_motive.**AndersonToDrinfeld**(*parent*, *phi*)

Bases: Map

The canonical isomorphism $M(\phi) \to K\{\tau\}$ for a Drinfeld module $\phi : A \to K\{\tau\}$.

class sage.rings.function_field.drinfeld_modules.anderson_motive.**DrinfeldToAnderson**(*parent*, *phi*)

Bases: Map

The canonical isomorphism $K\{\tau\} \to M(\phi)$ for a Drinfeld module $\phi : A \to K\{\tau\}$.

CHAPTER

TWO

INDICES AND TABLES

- Index
- Module Index
- Search Page

PYTHON MODULE INDEX

r

sage.rings.function_field.drinfeld_modules.anderson_motive, 1

INDEX

A

AndersonMotive() (*in module sage.rings.function_field.drinfeld_modules.anderson_motive*), 7
AndersonMotive_drinfeld (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 10
AndersonMotive_general (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 11
AndersonMotive_homspace (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 13
AndersonMotiveMorphism (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 9
AndersonQuotientMotive (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 13
AndersonSubMotive (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 13
AndersonToDrinfeld (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 13

C

characteristic_polynomial() (*sage.rings.function_field.drinfeld_modules.anderson_motive.AndersonMotiveMorphism method*), 9
charpoly() (*sage.rings.function_field.drinfeld_modules.anderson_motive.AndersonMotiveMorphism method*), 10

D

drinfeld_module() (*sage.rings.function_field.drinfeld_modules.anderson_motive.AndersonMotive_drinfeld method*), 11
DrinfeldToAnderson (*class in sage.rings.function_field.drinfeld_modules.anderson_motive*), 13

E

Element (*sage.rings.function_field.drinfeld_modules.anderson_motive.AndersonMotive_homspace attribute*), 13

H

hodge_pink_weights() (*sage.rings.function_field.drinfeld_modules.anderson_motive.AndersonMotive_general method*), 11

I

is_effective() (*sage.rings.function_field.drinfeld_modules.anderson_motive.AndersonMotive_general method*), 12

M

module
    sage.rings.function_field.drinfeld_modules.anderson_motive, 1

S

sage.rings.function_field.drinfeld_modules.anderson_motive
    module, 1