

The Origami Package

Computing Veechgroups of Origamis

Version 1.0.0

13.11.2018

Pascal Kattler
Andrea Thevis

Pascal Kattler Email: kattler@math.uni-sb.de
Homepage: <http://www.math.uni-sb.de/ag/weitze/>
Address: AG Weitze-Schmithüsen
FR 6.1 Mathematik
Universität des Saarlandes
D-66041 Saarbrücken

Andrea Thevis Email: thevis@math.uni-sb.de
Homepage: <http://www.math.uni-sb.de/ag/weitze/>
Address: AG Weitze-Schmithüsen
FR 6.1 Mathematik
Universität des Saarlandes
D-66041 Saarbrücken

Copyright

© 2018 by Pascal Kattler

Acknowledgements

We thank Sergio Siccha for his support and valuable input at the beginning of this project. Further, we would like to thank Vincent Delecroix and Samuel Lelièvre for fruitful discussions about algorithms implemented in the `surface_dynamics` package in `sage`. Some of the functionality of the `surface_dynamics` package can be used in this package as well. For this we use an interface between `GAP` and `sage`. We are thankful to Mohamed Barakat and Markus Pfeiffer for helping using the interface. Moreover, we thank Thomas Breuer for helpful ideas and comments on some algorithms implemented in this package. Supported by Project I.8 of SFB-TRR 195 'Symbolic Tools in Mathematics and their Application' of the German Research Foundation (DFG).

Contents

1	The Veechgroup of Origamis	4
1.1	Introduction	4
1.2	The Free Group	4
1.3	The Origami Object	4
1.4	List of Origamis	7
1.5	Using SageMath functions	8
	References	12

Chapter 1

The Veechgroup of Origamis

1.1 Introduction

This package provides calculations with origamis. An origami (also known as square-tiled surface) is a finite covering of a torus which is ramified at most over one point. It can be described in the following way from two permutations $\sigma_x, \sigma_y \in S_d$. We take d squares Q_1, \dots, Q_d and glue the lower side of Q_i to the upper side of $Q_{\sigma_y(i)}$ and the right side of Q_i to the left side of $Q_{\sigma_x(i)}$. We require origamis to be connected and thus the group generated by σ_x and σ_y acts transitively on $\{1, \dots, d\}$. In this package we identify an origami with a pair of permutations, which acts transitively on $\{1, \dots, d\}$ up to simultaneous conjugation. We introduce a new type of origamis, namely origami objects, which are created by this two permutations and its degree. The degree of an origami is the number of squares. Origamis are stored as such objects. We are mainly interested in the Veechgroup of an origami. It can be shown that the Veechgroup of an origami is a subgroup of $SL_2(\mathbb{Z})$ of finite index. So we fix two generators of $SL_2(\mathbb{Z})$

$$S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

and

$$T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

1.2 The Free Group

In this package we fix the free group F generated by \tilde{S} and \tilde{T} . We consider the canonical epimorphism $\pi : F \rightarrow SL_2(\mathbb{Z})$ with $\pi(\tilde{S}) = S$ and $\pi(\tilde{T}) = T$.

1.3 The Origami Object

In this section we describe the main functions of this package.

1.3.1 Origami

▷ `Origami(permX, permy, d)`

(function)

This function generates a new origami object with $\sigma_x = \text{permX}$, $\sigma_y = \text{permY}$ and degree d .

Example

```
gap> Origami((1,2), (2,3), 3);
Origami((1,2), (2,3), 3)
```

▷ `OrigamiWithoutTest(permX, permy, d)` (function)

This function does the same as `Origami`, but in contrast it does not test, whether the origami describes a connected surface. ▷ `ActionOfSpecialLinearGroup(matrix, origami)` (operation)

The group $SL_2(\mathbb{Z})$ acts on the set of origamis of fixed degree. The following methods for this operation are installed. ▷ `ActionOfSpecialLinearGroup(word, origami)` (method)

Given a word $word$ in the free group $Group(\tilde{S}, \tilde{T})$ this function computes $\pi(word) \in SL_2(\mathbb{Z})$ and returns $\pi(word).origami$. The word is given as a string, as shown in the following example.

Example

```
gap> ActionOfSpecialLinearGroup("ST", Origami((1,3,5), (1,3)(2,4,5), 5));
Origami((1,3)(2,5,4), (2,4,5,3), 5)
```

▷ `ActionOfSpecialLinearGroup(matrix, origami)` (method)

Given matrix in $SL_2(\mathbb{Z})$ this function returns $matrix.origami$. The word is given as a string, as shown in the following example.

Example

```
gap> ActionOfSpecialLinearGroup([ [ 0, -1 ], [ 1, 1 ] ], Origami((1,3,5),
>(1,3)(2,4,5), 5));
Origami((1,3)(2,5,4), (2,4,5,3), 5)
```

▷ `ActionOfF2ViaCanonical(origami, word)` (function)

Given a word $word$ in the free group $Group(\tilde{S}, \tilde{T})$ this function computes $\pi(word) \in SL_2(\mathbb{Z})$ and returns $\pi(word).origami$. But in contrast to `ActionOfSpecialLinearGroup` the result is stored in the canonical representation. ATTENTION: the order of arguments is switched compared to the order of the arguments in the function `ActionOfSpecialLinearGroup`.

Example

```
gap> ActionOfF2ViaCanonical(Origami((1,2), (1,3), 3), "S");
Origami((1,2), (2,3), 3)
```

▷ `RightActionOfF2ViaCanonical(origami, word)` (function)

This function computes the right action of the projection of a word $word$ in the free group $Group(\tilde{S}, \tilde{T})$ on an origami $origami$. It returns $origami.\pi(word) = \pi(word)^{-1}.origami$, where the left action is the common action of $SL_2(\mathbb{Z})$ on origamis of a given degree. This action has the same orbits as the left action. For the Veechgroup computation both actions can be used and give the same result. In contrast to `ActionOfSpecialLinearGroup` the result is stored in the canonical representation. ATTENTION: the order of arguments is switched compared to the order of the arguments in the function `ActionOfSpecialLinearGroup`.

Example

```
gap> RightActionOffF2ViaCanonical(Origami((2,3), (1,3,2), 3), "T");
Origami((1,2), (2,3), 3)
```

▷ `OrigamiNormalForm(origami)` (function)

This function calculates a canonical representation of *origami*. Two origamis are equivalent, if they have the same canonical representation. This function has been implemented by Luca Junk and the algorithm is from [McK].

Example

```
gap> OrigamiNormalForm(Origami((1,3,5,7)(2,8,4,10)(6,9), (1,5,2,6,3)(4,10)(7,9), 10));
Origami((1,2)(3,5,4,6)(7,8,9,10), (1,3)(2,4,5,6,7)(9,10), 10)
```

▷ `OrigamiFamily` (family)

Since origamis do not fit in any existing family in GAP, we introduce a new family for origami objects called `OrigamiFamily`. ▷ `HorizontalPerm(origami)` (attribute)

This function returns the horizontal permutation σ_y of an origami *origami*.

Example

```
gap> HorizontalPerm(Origami((1,3,5), (1,3)(2,4,5), 5));
(1,3,5)
```

▷ `VerticalPerm(origami)` (attribute)

This function returns the vertical permutation σ_x of an origami *origami*.

Example

```
gap> VerticalPerm(Origami((1,3,5), (1,3)(2,4,5), 5));
(1,3)(2,4,5)
```

▷ `DegreeOrigami(origami)` (attribute)

This function returns the degree of an *origami*.

Example

```
gap> DegreeOrigami(Origami((1,3,5), (1,3)(2,4,5), 5));
5
```

▷ `Stratum(origami)` (attribute)

This function calculates the stratum of an origami *origami*. The stratum of an origami is a list of the nonzero degrees of the singularities. For a singularity with cone angle $2\pi k$ the degree of the singularity is $k - 1$.

Example

```
gap> Stratum(Origami((1,6,4,7,5,3)(2,8), (1,4,5,3,8,2,6), 8));
[ 1, 5 ]
```

▷ `Genus(origami)` (attribute)

This function calculates the genus of the origami surface.

Example

```
gap> Genus( Origami((1,2,3,4),(1,2)(3,4), 4) );
2
```

▷ `VeechGroup(origami)`

(attribute)

This function calculates the Veechgroup of an origami *origami*. The Veechgroup G of *origami* is a subgroup $SL_2(\mathbb{Z})$ of finite index. The group is stored as a `ModularSubgroup` from the `ModularSubgroup` package. It is represented by two permutations σ_S and σ_T describing how the generators S and T of $SL_2(\mathbb{Z})$ act on the cosets of G in $SL_2(\mathbb{Z})$. E.g, if $SH_i = H_j$ and H_i, H_j are the cosets associated to the integers i, j , respectively, then $\sigma_S(i) = j$. The algorithm was introduced by Gabriela Weitzschmuth in [Sch04]. You get the coset permutations using the `ModularSubgroup` package as in the following example.

Example

```
gap> SAction(VeechGroup(Origami((1,2,5)(3,4,6), (1,2)(5,6), 6)));
(1,3)(2,5)(4,7)(6,8)(9,10)
gap> TAction(VeechGroup(Origami((1,2,5)(3,4,6), (1,2)(5,6), 6)));
(1,2,4)(3,6)(5,8,7,9,10)
```

▷ `Cosets(origami)`

(attribute)

This function calculates the right cosets of the Veechgroup of an origami *origami* as a list of words in S and T .

Example

```
gap> Cosets(Origami((1,2,5)(3,4,6), (1,2)(5,6), 6));
[ <identity ...>, S, T, T^-1, S*T, S*T^-1, T*S, T^-1*S, S*T*S, S*T^-1*S ]
```

▷ `EquivalentOrigami(origami1, origami2)`

(function)

This function tests whether *origami1* is equal to *origami2* up to renumbering of the squares.

Example

```
gap> EquivalentOrigami(Origami((1,4)(2,6,3), (1,5)(2,3,6,4), 6), Origami((1,4,3)
>(2,5), (1,5,3,4)(2,6), 6));
true

gap> EquivalentOrigami(Origami((1,4)(2,6,3), (1,5)(2,3,6,4), 6), Origami((1,2,5)
>(3,4,6), (1,2)(5,6), 6));
false
```

1.4 List of Origamis

The following functions generate lists of Origamis.

1.4.1 OrigamiList

▷ `OrigamiList(d)`

(function)

This Function calculates a list of all origamis, with a given degree d .

Example

```
gap> OrigamiList(2);
[ Origami((), (1,2), 2), Origami((1,2), (), 2), Origami((1,2), (1,2), 2) ]
```

▷ `OrigamiListWithStratum(d, stratum)`

(function)

This function calculates a list of all origamis with a given degree *d* and stratum *stratum*.

Example

```
gap> OrigamiListWithStratum(5, [1,1]);
[ Origami((1,2), (1,3)(2,4,5), 5), Origami((1,2), (1,3,2,4,5), 5),
  Origami((1,2,3), (1,2)(3,4,5), 5), Origami((1,2,3), (2,3,4,5), 5),
  Origami((1,2,3), (2,4,5,3), 5), Origami((1,2)(3,4,5), (2,3), 5),
  Origami((1,2)(3,4,5), (1,2,3), 5), Origami((1,2)(3,4,5), (2,3,4,5), 5),
  Origami((1,2)(3,4,5), (2,3,5,4), 5), Origami((1,2)(3,4,5), (1,2,3,4,5), 5),
  Origami((1,2)(3,4,5), (1,2,3,5,4), 5), Origami((1,2,3,4), (3,4,5), 5),
  Origami((1,2,3,4), (3,5,4), 5), Origami((1,2,3,4), (1,2,3)(4,5), 5),
  Origami((1,2,3,4), (1,3,2)(4,5), 5), Origami((1,2,3,4), (2,3,4,5), 5),
  Origami((1,2,3,4), (2,5,4,3), 5), Origami((1,2,3,4), (1,2,4,5,3), 5),
  Origami((1,2,3,4), (1,3,5,4,2), 5), Origami((1,2,3,4,5), (3,5), 5),
  Origami((1,2,3,4,5), (1,2)(3,4,5), 5), Origami((1,2,3,4,5), (1,2)(3,5,4), 5),
  Origami((1,2,3,4,5), (2,4,3,5), 5), Origami((1,2,3,4,5), (2,5,3,4), 5) ]
```

1.5 Using SageMath functions

The SageMath Package `surface_dynamics` from Vincent Delecroix [VD] provides methods for origamis. To use the functions of this chapter, Sage must be installed on your operation system. The following descriptions and examples are mainly taken from the manual of [VD].

1.5.1 VeechgroupBySage

▷ `VeechgroupBySage(origami)`

(function)

This function executes the SageMath method `veech_group` to *origami* and returns its result as GAP object. It does the same as `VeechGroup` (1.3.1).

Example

```
gap> VeechgroupBySage(Origami((1,3,5), (1,3)(2,4,5), 5));
<modular subgroup of index 24>
```

▷ `NormalFormBySage(origami)`

(function)

This function executes the SageMath method `to_standard_form` to *origami* and returns its result as GAP object. In principle it makes the same as `OrigamiNormalForm` (1.3.1), but it has an other representation.

Example

```
gap> NormalFormBySage(Origami((1,3,5), (1,3)(2,4,5), 5));
Origami((3,4,5), (1,2,3)(4,5), 5)
```


▷ `IsHyperellipticBySage(origami)` (function)

This function executes the SageMath method `IsHyperelliptic` to *origami* and returns its result as GAP object. It tests, whether *origami* is hyperelliptic.

Example

```
gap> IsHyperellipticBySage(Origami((1,3), (1,2), 3));
true
```

▷ `IsPrimitiveBySage(origami)` (function)

This function executes the SageMath method `IsPrimitive` to *origami* and returns its result as GAP object. An origami is primitive if it does not cover another origami. An origami is primitive if the action of the monodromy group has no non-trivial block.

Example

```
gap> IsPrimitiveBySage(Origami((1,3), (1,2), 3));
true
gap> IsPrimitiveBySage(Origami((1,2)(3,4), (1,3,5,6)(2,4), 6));
false
```

▷ `ReduceBySage(origami)` (function)

This function executes the SageMath method `Reduce` to *origami* and returns its result as GAP object. It returns a reduced origami isomorphic (up to $SL(2, \mathbb{Q})$ action) to that origami.

Example

```
gap> ReduceBySage(Origami((1,2)(3,4), (1,3,5,6)(2,4), 6));
Origami((1,2), (1,3), 6)
```

▷ `AbsolutePeriodGeneratorsBySage(origami)` (function)

This function executes the SageMath method `AbsolutePeriodGenerators` to *origami* and returns its result as GAP object. It returns a generating set of the absolute periods of this origami. To each curve on an origami, we can associate its holonomy (that is an element of $\mathbb{Z}\mathbb{Z} \times \mathbb{Z}\mathbb{Z}$). This function returns a generating set of the module generated by holonomies of closed curves.

Example

```
gap> AbsolutePeriodGeneratorsBySage(Origami((1,2,3,4)(5,6), (1,5)(2,6), 6));
[ [ 2, 0 ], [ 2, 0 ], [ 0, 1 ], [ 0, 1 ] ]
```

▷ `LatticeOfAbsolutePeriodsBySage(origami)` (function)

This function executes the SageMath method `LatticeOfAbsolutePeriods` to *origami* and returns its result as GAP object. It returns (a, t, u) where $((a, 0), (t, u))$ is a standard basis for the lattice of the absolute periods of self. The lattice of periods of an origami is the sublattice of \mathbb{Z}^2 generated by the holonomy vectors of its saddle connections. Any sublattice of \mathbb{Z}^2 has a standard basis consisting of a horizontal vector $(a, 0)$ and a nonhorizontal vector (t, u) , where a, t, u are integers satisfying $0 < t < a$ and $0 < t$.

Example

```
gap> LatticeOfAbsolutePeriodsBySage(Origami((1,2)(3,4), (2,3), 4));
[ [ 2, 0 ], [ 0, 1 ] ]
```

▷ `LatticeOfQuotientsBySage(origami)` (function)

This function executes the SageMath method `LatticeOfQuotients` to *origami* and returns its result as GAP object. It return the lattice of quotients of this origami. The set of quotients of an origami contain a maximal element (itself) and a minimal element (the 1-torus). More generally, it is organised as a lattice. Not implemented.

Example

▷ `OptimalDegreeBySage(origami)` (function)

This function executes the SageMath method `OptimalDegree` to *origami* and returns its result as GAP object. The optimal degree of self is the degree of the map to the largest torus. Any origami $X \Rightarrow T$ factor as $i \circ \pi_{opt}$ where i is an isogeny. The optimal degree is the degree of π_{opt} .

Example

```
gap> OptimalDegreeBySage(Origami((1,2)(3,4), (2,3), 4));
2
```

▷ `PeriodGeneratorsBySage(origami)` (function)

This function executes the SageMath method `PeriodGenerators` to *origami* and returns its result as GAP object. It return a list of periods that generate the lattice of periods.

Example

```
gap> PeriodGeneratorsBySage(Origami((1,3,6)(2,5,7)(4), (1,2,4,3,5,6,7), 7));
[ [ 2, 0 ], [ 1, 0 ], [ 1, 0 ], [ 0, 2 ], [ -1, 2 ], [ 0, 1 ] ]
```

▷ `WidthsAndHeightsBySage(origami)` (function)

This function executes the SageMath method `WidthsAndHeights` to *origami* and returns its result as GAP object. It return the list of widths and heights of cylinder.

Example

```
gap> WidthsAndHeightsBySage(Origami((1,2)(3,4), (1,3,5,6)(2,4), 4));
[ [ 1, 2 ], [ 2, 2 ] ]
```

▷ `SumOfLyapunovExponentsBySage(origami)` (function)

This function executes the SageMath method `SumOfLyapunovExponents` to *origami* and returns its result as GAP object. It returns the sum of Lyapunov exponents for this origami.

Example

```
gap> SumOfLyapunovExponentsBySage(Origami((1,2)(3,4), (2,3), 4));
3/2
```

▷ `LyapunovExponentsApproxBySage(origami)` (function)

This function executes the SageMath method `LyapunovExponentsApprox` to *origami* and returns its result as GAP object. Not implemented.

Example

▷ `IntermediateCoversBySage(origami)` (function)

This function executes the SageMath method `NormalFormBySage` to *origami* and returns its result as **GAP** object. It return the list of intermediate covers of this origami. Not implemented

Example

References

- [McK] Brendan McKay. The simultaneous conjugacy problem in the symmetric group s_n . [6](#)
- [Sch04] Gabriela Schmithüsen. An algorithm for finding the Veech group of an origami. *Experiment. Math.*, 13(4):459–472, 2004. [7](#)
- [VD] Samuel Lelièvre Vincent Delecroix, Charles Fougerson. `surface_dynamicssagemathpackageversion0.4.0..`
[8](#)