JOHANNES KEPLER
UNIVERSITÄT LINZ

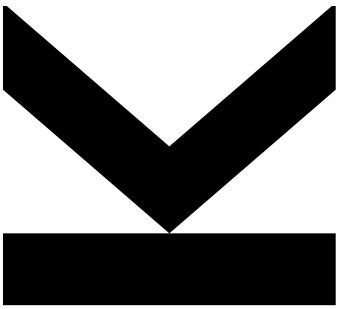
Submitted by
Marcus Silvester Aichmayr

Submitted at
Institute for Algebra

Supervisor
Priv.-Doz.
Dr. Georg Regensburger

August 2021

Computing with sign vectors in SageMath and applications to exponential maps



Master's thesis
to obtain the academic degree of
Diplom-Ingenieur
in the Master's Program
Computer mathematics

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, 3. August 2021
Ort, Datum

Marcus Aschmayr
Unterschrift

Zusammenfassung

Diese Masterarbeit verbindet die Theorie orientierter Matroide mit der Theorie reeller Vektorräume. Um einen Vorzeichenvektor zu erhalten, wird die Vorzeichenfunktion komponentenweise auf einen reellen Vektor angewandt. Wird die Vorzeichenfunktion auf jeden Vektor eines reellen Vektorraums angewandt, erhält man einen orientierten Matroiden. Viele Eigenschaften reeller Vektoren wie Trägermenge oder Orthogonalität lassen sich analog für Vorzeichenvektoren motivieren.
Im Rahmen dieser Arbeit wurden zwei Pakete für das Computeralgebrasystem SageMath entwickelt. Das erste Paket beinhaltet Operationen für Vorzeichenvektoren und Elementarvektoren. Außerdem enthält es einige Funktionen, um mit orientierten Matroiden zu rechnen. Das zweite Paket behandelt generalisierte polynomiale und exponentielle Abbildungen. Viele Bedingungen zur Bijektivität solcher Abbildungen können algorithmisch überprüft werden. Die entsprechenden Befehle sind im Paket verfügbar.
Ein wesentlicher Teil dieser Arbeit befasst sich mit der Überprüfung einer Nicht-Degeneriertheitsbedingung reeller Vektorräume. Damit eine exponentielle Abbildung als bijektiv klassifiziert werden kann, müssen die repräsentierenden reellen Vektorräume nicht-degeneriert sein. Zusätzlich wird ein Algorithmus zur Überprüfung dieser Bedingung beschrieben. Dieser Algorithmus ist im zweiten Paket implementiert.

Abstract

This Master's thesis connects the theory of oriented matroids to the theory of real vector spaces. By applying the sign function component-wise to a real vector, we obtain a sign vector. If we do this for each vector of a real subspace, we obtain an oriented matroid. Many properties, like support or orthogonality, can be motivated for sign vectors in an analogous way.
In the frame of this thesis, two packages for SageMath were developed. The first package deals with operations for sign vectors and elementary vectors. It also offers several functions to compute with oriented matroids. The second package is about generalized polynomial and exponential maps. Several conditions for injectivity and bijectivity of such maps can be checked algorithmically and are offered in the package as functions.
A major part of this thesis is the study of a certain non-degeneracy condition of real subspaces representing an exponential map. As part of this thesis, an algorithm to check this condition was developed and implemented in the second package.

Eidesstattliche Erklärung i

Zusammenfassung ii

Abstract iii

Contents iv

1 Introduction 1

2 Elementary vectors 2
2.1 A fundamental Theorem by R. T. Rockafellar . . . 3
2.2 Minty's Lemma . . . 4

3 Sign vectors 7
3.1 Basic operations and definitions . . . 7
3.2 Support . . . 8
3.3 Composition . . . 9
3.4 Separating elements . . . 10
3.5 Conformal relation . . . 11
3.6 Orthogonality . . . 13
3.7 Closure . . . 14
3.8 Positive and negative sign vectors . . . 16

4 Oriented matroids 17
4.1 Cocircuits . . . 22
4.2 Topes . . . 25
4.3 Face enumeration . . . 29
4.3.1 Parallelness . . . 29
4.3.2 Rank and dimension . . . 33
4.3.3 Deletion . . . 34
4.3.4 Contraction . . . 35
4.3.5 Lower Faces . . . 37
4.3.6 Examples . . . 39
4.4 Duality . . . 42

5 Applications to bijectivity of exponential maps 45
5.1 General definitions . . . 45
5.2 Characterizing injectivity . . . 46
5.2.1 Examples . . . 47
5.3 Characterizing bijectivity . . . 50
5.3.1 Non-degenerate . . . 51
5.3.2 Examples . . . 53
5.4 Characterizing robust bijectivity . . . 56
5.4.1 Examples . . . 57

A Computing elementary vectors over integral domains 59

B Lattice theory 60
B.1 Partially ordered sets . . . 60
B.2 Lattices . . . 61

C Implementation **62**
C.1 Elementary vectors . . . 62
C.2 Sign vectors . . . 64
C.2.1 Class `SignVector` . . . 64
C.2.2 Functions for generating sign vectors . . . 65
C.2.3 Functions for lists of sign vectors . . . 65
C.2.4 Oriented matroids . . . 67
C.3 Bijectivity of exponential maps . . . 70
C.3.1 Functions . . . 70
C.3.2 Conditions . . . 71

References **74**

Index **76**

1 Introduction

This Master's thesis connects the theory of oriented matroids to the theory of real vector spaces. By applying the sign function component-wise to a real vector, we obtain a sign vector. If we do this for each vector of a real subspace, we obtain an oriented matroid. Many properties, like support or orthogonality, can be motivated for sign vectors in an analogous way.
In the framework of this thesis, two packages for SageMath [Sage] were developed. The first package [`sign_vectors`] deals with operations for sign vectors and elementary vectors. Several algorithms proposed in [Fin01] and the algorithm from [FST91] to compute with oriented matroids are also implemented in this package. The second package [`bijectivity_exponential_maps`] is about generalized polynomial and exponential maps. Several conditions for properties of such maps from [MHR19] can be checked algorithmically and are therefore offered as functions in the package.

In Chapter 2, we summarize some important results about elementary vectors. See also [MR16] and [Aic20]. Chapter 3 deals with definitions of sign vectors.
Oriented matroids are a major part of this thesis, represented in Chapter 4. An oriented matroid can be defined by axioms that closely relate it to a vector space.
Similar to elementary vectors, the cocircuits are the sign vectors with minimal support, that is, a maximal number of zero entries. These elements can be used to determine the whole oriented matroid. The corresponding algorithm can found in [Fin01] and is implemented in [`sign_vectors`]. The package also offers an algorithm to compute the cocircuits from a real vector space by computing the elementary vectors. Thus, the package yields a revised version of the corresponding implementation initially found in [Aic20].
On the other hand, topes are the elements with maximal support. Those sign vectors can be determined algorithmically as well. The respective algorithm from [Fin01] is also implemented in the package.
An algorithm from [FST91] constructs an oriented matroid by using the tope set. Here, the covectors are separated by their rank. The package also offers an implementation of this algorithm.
The final chapter is about polynomial and exponential maps. Here, we are interested whether such a map is injective, bijective or bijective with respect to small perturbations. Several results from [MHR19] can be checked algorithmically. The corresponding conditions are implemented in the package [`bijectivity_exponential_maps`].
A major part of this theses is the study of a certain non-degeneracy condition of real subspaces representing an exponential map. As part of this thesis, an algorithm to check this condition has been developed. The package [`bijectivity_exponential_maps`] offers an implementation of this algorithm.

Appendix A states results about the computation of elementary vectors. For further details, see [Aic20]. Lattice theory plays an important role for oriented matroids and is therefore summarized in appendix B. Finally, appendix C lists the documentations of the packages [`sign_vectors`] and [`bijectivity_exponential_maps`].

2 Elementary vectors

In this section, we summarize some important results about elementary vectors. See also [MR16] and [Aic20] for further details and references.

In this chapter, we denote by $\mathcal{R}$ an integral domain.

Definition 2.1. Let $x \in \mathcal{R}^n$ be a vector. By

$$\underline{x} := \{k \in \{1, \dots, n\} \mid x_k \neq 0\},$$

we denote the *support* of the vector x.

Next, we will introduce elementary vectors. These are non-zero vectors that have a maximal number of zero entries.

Definition 2.2. Let $\mathcal{A}$ be a subset of $\mathcal{R}^n$ We call a non-zero vector $x \in \mathcal{A}$ *support-minimal*, if for all non-zero $y \in \mathcal{A}$, we have

$$\underline{y} \subseteq \underline{x} \quad \text{implies} \quad \underline{y} = \underline{x}.$$

A support-minimal vector is also called *elementary vector*.

Typically, one introduces elementary vectors of subspaces. Note that there are only finitely many elementary vectors up to multiples. A subspace can always be written as the kernel of a matrix. Therefore, we consider an algorithm to compute elementary vectors in the kernel of a given real matrix. As it turns out, we can do the same if we have a matrix over an integral domain. This is because we can use a commutative version of Cramer's rule to compute each elementary vector and in that way, we do not need division. The corresponding results are formalized in Appendix A.
The initial implementation of the corresponding algorithm can be found in [Aic20]. The package [`elementary_vectors`] offers a revised version of this implementation. After installing the package, we can load the function:

```
sage: from elementary_vectors import elementary_vectors
```

Example 2.3. We consider the following matrix:

```
sage: M = matrix([[1,0,2,1],[0,1,1,2]])
sage: M
[1 0 2 1]
[0 1 1 2]
```

Now, we apply the function `elementary_vectors` to M.

```
sage: elementary_vectors(M, kernel=True)
[(-2, -1, 1, 0), (-1, -2, 0, 1), (3, 0, -2, 1), (0, 3, 1, -2)]
```

Hence, we obtain the elementary vectors lying in the kernel of M. By passing the argument `kernel=False`, we can compute the elementary vectors determined by the row space of M.

```
sage: elementary_vectors(M, kernel=False)
[(2, -1, 3, 0), (-1, 2, 0, 3), (1, 0, 2, 1), (0, 1, 1, 2)]
```

Note that the argument `kernel=` is optional. By default, `elementary_vectors` returns the elementary vectors in the row space.

```
sage: elementary_vectors(M)
[(2, -1, 3, 0), (-1, 2, 0, 3), (1, 0, 2, 1), (0, 1, 1, 2)]
```

Example 2.4. The next matrix has higher dimension.

```
sage: M = matrix([[1,0,0,0,1],[0,1,0,-1,-2],[0,0,1,1,2]])
sage: M
[ 1  0  0  0  1]
[ 0  1  0 -1 -2]
[ 0  0  1  1  2]
```

We apply `elementary_vectors` and compute the elementary vectors in the kernel of M.

```
sage: elementary_vectors(M, kernel=True)
[(0, -1, 1, -1, 0), (1, -2, 2, 0, -1), (1, 0, 0, 2, -1)]
```

Here, we have 3 elementary vectors, up to multiples.
Now, we compute the elementary vectors in the row space of M.

```
sage: elementary_vectors(M)
[(0, 1, 1, 0, 0), (-2, -1, 0, 1, 0), (1, 0, 0, 0, 1), (2, 0, -1, -1, 0),
   (0, 1, 0, -1, -2), (0, 0, 1, 1, 2)]
```

Therefore, the row space of the matrix M contains 6 elementary vectors, up to multiples.

2.1 A fundamental Theorem by R. T. Rockafellar

In this section, we discuss sign vectors of real vectors. Further details are presented in Chapter 3.

Definition 2.5. For a real number $a \in \mathbb{R}$, we define

$$\operatorname{sign}(a) := \begin{cases} - & \text{if } a < 0, \\ 0 & \text{if } a = 0, \\ + & \text{if } a > 0 \end{cases}$$

as the *sign* of a.

Let $x \in \mathbb{R}^n$ be a vector. The *sign vector*

$$\operatorname{sign}(x) \in \{-, 0, +\}^n$$

is defined by applying sign component-wise to x. Therefore, we have $\operatorname{sign}(x)_i = \operatorname{sign}(x_i)$ for $i = 1, \ldots, n$.

Frequently, we will apply sign to subsets of $\mathbb{R}^n$. Here, we mean a set of sign vectors that is obtained by applying sign to each vector of the corresponding subset.

Definition 2.6. For a subset $\mathcal{A}$ of $\mathbb{R}^n$, we define

$$\operatorname{sign}(\mathcal{A}) := \{\operatorname{sign}(x) \mid x \in \mathcal{A}\} \subseteq \{-, 0, +\}^n.$$

Next, we introduce a partial order on a set of sign vectors.

Definition 2.7. Let $X, Y \in \{-, 0, +\}^n$ be two sign vectors. We say X *conforms to* Y (denoted by $X \preceq Y$) if $X_k \neq 0$ implies $X_k = Y_k$ for every $k = 1, \ldots, n$.

Further details and basic notions about partially ordered sets are summarized in Appendix B.

By the following theorem from R. T. Rockafellar, see [Roc69], we can represent every vector of a subspace as a conformal sum of elementary vectors, that is, in no coordinate, two summands have opposing signs.

Theorem 2.8. *Let $\mathcal{V}$ be a subspace of the real vector space $\mathbb{R}^n$ and let $x \in \mathcal{V}$ be a non-zero vector. Then, there is a set $\mathcal{A} \subseteq \mathcal{V}$ of elementary vectors with*

$$x = \sum_{v \in \mathcal{A}} v$$

such that $\operatorname{sign}(v) \preceq \operatorname{sign}(x)$ *for all* $v \in \mathcal{A}$.

For the proof, see for example [MR16] and [Aic20].

2.2 Minty's Lemma

Let $\mathcal{V}$ be a subspace of $\mathbb{R}^n$ and let $I_k \subseteq \mathbb{R}$ be non-empty real intervals for $k = 1, \ldots, n$. The following theorem lets us use elementary vectors of the orthogonal complement $\mathcal{V}^\perp$ to check whether there exists a vector $x \in \mathcal{V}$ with $x_k \in I_k$ for all $k = 1, \ldots, n$. For an elementary proof, we refer to [Min74].

Theorem 2.9. *Let $\mathcal{V}$ be a subspace of $\mathbb{R}^n$ and let $I_k \subseteq \mathbb{R}$ be non-empty real intervals for $k = 1, \ldots, n$. A vector $x \in \mathcal{V}$ with $x_k \in I_k$ for $k = 1, \ldots, n$ exists if and only if for each elementary vector $v \in \mathcal{V}^\perp$, we have*

$$0 \in \left\{ \sum_{k=1}^{n} z_k v_k \mid z_k \in I_k \quad \text{for all } k \in \{1, \ldots, n\} \right\}. \tag{2.1}$$

Theorem 2.9 can also be formulated as a theorem of the alternative. In particular, an elementary vector of $\mathcal{V}^\perp$ can be used as certificate that there does not exist such a vector $x \in \mathcal{V}$. See also Theorem 22.1 in [Roc70].

Theorem 2.10. *Let $\mathcal{V}$ be a subspace of $\mathbb{R}^n$ and let $I_k \subseteq \mathbb{R}$ for $k = 1, \ldots, n$ be non-empty real intervals. Then, exactly one of the following alternatives holds:*

1. *There exists a vector $x \in \mathcal{V}$ such that $x_k \in I_k$ for $k = 1, \ldots, n$.*
2. *There exists an elementary vector $v \in \mathcal{V}^\perp$ such that*

$$0 < \sum_{k=1}^{n} z_k v_k \quad \text{for all } z_k \in I_k \text{ and } k = 1, \ldots, n.$$

Theorem 2.10 is often referred to as Minty's Lemma. Since there are only finitely many elementary vectors, up to multiples, equation (2.1) can be checked algorithmically. A corresponding implementation in Sage can already be found in [Aic20]. There is a revised version provided by the package [`elementary_vectors`]. We load this function:

```
sage: from elementary_vectors import exists_vector
```

Example 2.11. First, we consider the vector space of all vectors $v \in \mathbb{R}^3$ whose first two coordinates are equal and the last coordinate is zero. This vector space can be described as row space of the following matrix:

```
sage: M = matrix([1,1,0])
```

```
sage: M
[1 1 0]
```

To apply Theorem 2.9, we define the intervals $I_1 := [2, 5]$, $I_2 := [5, 6]$ and $I_3 := [-1, 1]$. To pass those intervals to `exists_vector`, we construct two lists.

```
sage: L = [2,5,-1]
sage: R = [5,6,1]
```

The list `L` contains the lower interval bounds and the list `R` consists of the upper interval bounds. Now, we apply the function:

```
sage: exists_vector(M, L, R)
True
```

The result is not surprising. For instance, the vector $(5, 5, 0)$ is a solution.

We can consider open intervals $I_1 := (2, 5)$, $I_2 := (5, 6)$ and $I_3 := (-1, 1)$ if we pass `l=False` and `r=False` to `exists_vector`.

```
sage: exists_vector(M, L, R, l=False, r=False)
False
```

By passing only `r=False`, we obtain mixed intervals $I_1 := [2, 5)$, $I_2 := [5, 6)$ and $I_3 := [-1, 1)$.

```
sage: exists_vector(M, L, R, r=False)
False
```

If the result of `exists_vector` is `False`, there is an elementary vector that acts as a certificate. To obtain this elementary vector, we specify `certificate=True`.

```
sage: exists_vector(M, L, R, r=False, certificate=True)
[False, (1, -1, 0)]
```

To take arbitrary intervals, we can also pass lists of booleans to `l` and `r`. For instance, if we have the intervals $I_1 := (2, 5]$, $I_2 := [5, 6)$ and $I_3 := (-1, 1]$, we can choose `l` and `r` in the following way:

```
sage: l = [False,True,False]
sage: r = [True,False,True]
sage: exists_vector(M, L, R, l=l, r=r)
True
```

Similar to `elementary_vectors`, the function `exists_vector` supports the optional argument `kernel=True` to consider the kernel of the given matrix instead of the row space.

```
sage: exists_vector(M, L, R, kernel=True)
False
```

Example 2.12. Unbounded intervals are also supported by `exists_vector`.

```
sage: M = matrix([[1,0,1,0],[0,1,1,1]])
sage: L = [2,5,0,-oo]
sage: R = [5,oo,8,5]
```

```
sage: exists_vector(M, L, R)
True
```

Finally, we can also use a list of precomputed elementary vectors instead of a matrix.

```
sage: evs = elementary_vectors(M, kernel=True)
sage: exists_vector(evs, L, R)
True
```

3 Sign vectors

In this section, we discuss properties of sign vectors. Sign vectors are closely related to real vectors. Many properties, like support or orthogonality, can be motivated for sign vectors in an analogous way. As a main reference for this chapter, we use [Fin01]. See also [Zie95].

For the remaining chapters, let E be a finite set.

3.1 Basic operations and definitions

Definition 3.1. An element $X \in \{-, 0, +\}^E$ is called a *sign vector* on E. We define the zero sign vector $\mathbf{0} \in \{-, 0, +\}^E$ as the sign vector that has 0 at each component.

One can introduce sign vectors with components in $\{-1, 0, 1\}$. However, in order to not confuse sign vectors with real vectors, we define them with components in $\{-, 0, +\}$, where $-$ stands for -1 and $+$ stands for 1. Consequently, multiplying elements of $\{-, 0, +\}$ is defined analogously to multiplying elements of $\{-1, 0, 1\}$. Similarly, scalar multiplication is defined for sign vectors as well. To further distinguish sign vectors from real vectors, we omit the commas that separate the coordinates.

A class for working with sign vectors is implemented in the package [`sign_vectors`]. After installing the package, we can load it in the following way:

```
sage: from sign_vectors.sign_vectors import *
```

Example 3.2. The package offers several ways to construct sign vectors. The typical way is to use a list or a vector of real values. Here, a sign vector is obtained by applying the function `sign` component-wise.

```
sage: sign_vector([1,0,-1,1])
(+0-+)
sage: x = vector([5,2/5,-1,0])
sage: sign_vector(x)
(++-0)
```

An elegant way to construct a `SignVector` object is to use the sign pattern as a string.

```
sage: sign_vector('++-00+-')
(++-00+-)
```

There is also a constructor for the zero sign vector $\mathbf{0}$ of a given length.

```
sage: zero_sign_vector(6)
(000000)
```

Furthermore, we can use the package to construct random sign vectors.

```
sage: random_sign_vector(7)
(--+-0+-)
```

There are several operations defined for sign vectors.

Definition 3.3. Let X be a sign vector on E. The sign vector $-X$ is defined by $(-X)_e = -X_e$ for each $e \in E$.

Let $S \subseteq E$. By X_S, we denote the subvector of X on the set S. Similarly, $X \setminus S$ is the subvector that consists only of the components in $E \setminus S$.

Example 3.4. The operations from Definition 3.3 are implemented in the package.

```
sage: X = sign_vector('+++000--')
sage: X
(+++000--)
```

To reverse the signs of a sign vector, we apply the usual operator $-$.

```
sage: -X
(---000++)
```

Note that the first component of a `SignVector` object has the index 0, since this is standard in Python and therefore in Sage. Hence, when using the package, we will always consider sign vectors of length n with respect to the set

$$E = \{0, \ldots, n-1\}.$$

To construct the subvector X_S with $S = \{0, 2, 3\}$, we use the method `list_from_positions`.

```
sage: sign_vector(X.list_from_positions([0,2,3]))
(++0)
```

3.2 Support

As for real vectors, the support of a sign vector is the set of non-zero components. Furthermore, the support can be split into two parts depending on the sign of the corresponding entries.

Definition 3.5. The *support* of a sign vector X on E is the set

$$\underline{X} := \{e \in E \mid X_e \neq 0\}.$$

We define the *zero support* of X by

$$X^0 := \{e \in E \mid X_e = 0\}$$

and the *positive* and *negative support* by

$$X^+ := \{e \in E \mid X_e = +\}$$

and

$$X^- := \{e \in E \mid X_e = -\},$$

respectively.

Consequently, we obtain $X^0 = E \setminus \underline{X}$ and $X^+ \cup X^- = \underline{X}$.

Clearly, the support of a real vector, is the same as the support of the corresponding sign vector.

Example 3.6. Let X be the sign vector

$$X := (+\ +\ -\ 0\ -)$$

on $E = \{1, 2, 3, 4, 5\}$. Then, the different supports are

$$\underline{X} = \{1, 2, 3, 5\},\ X^0 = \{4\},\ X^+ = \{1, 2\} \text{ and } X^- = \{3, 5\}.$$

The package offers methods for the different types of support.

Example 3.7. We define a sign vector X and compute the different supports using Sage.

```
sage: X = sign_vector([-1,0,1,-1,0])
sage: X
(-0+-0)
sage: X.support()
[0, 2, 3]
sage: X.zero_support()
[1, 4]
sage: X.positive_support()
[2]
sage: X.negative_support()
[0, 3]
```

As for real vectors, there are sign vectors with minimal support.

Definition 3.8. Let $\mathcal{S} \subseteq \{-, 0, +\}^E$. A non-zero sign vector $X \in \mathcal{S}$ is called *support-minimal* if for all non-zero $Y \in \mathcal{S}$, we have

$$\underline{Y} \subseteq \underline{X} \quad \text{implies} \quad \underline{Y} = \underline{X}.$$

3.3 Composition

Composition of sign vectors is based on the following observation. Consider two vectors $x, y \in \mathbb{R}^E$. Now, we change x by adding a (small) positive multiple of y to x. In this case, the multiple is chosen such that non-zero entries of x keep the same sign. On the other hand, if a component of x is zero, then the sign changes to the sign of the corresponding element in y.

Definition 3.9. Let X and Y be two sign vectors on E. We define the *composition* $X \circ Y$ by

$$(X \circ Y)_e := \begin{cases} X_e & \text{if } X_e \neq 0, \\ Y_e & \text{otherwise,} \end{cases}$$

for each $e \in E$.

Example 3.10. We define the sign vectors

$$X = (+\ -\ 0\ ) \quad \text{and} \quad Y = (\ 0\ +\ +).$$

By Definition 3.9, we obtain

$$X \circ Y = (+\ -\ +) \quad \text{and} \quad Y \circ X = (+\ +\ +).$$

Note that in general composition is not commutative.

We use the package to compute the composition with Sage.

Example 3.11. We define two sign vectors X and Y and compute their compositions $X \circ Y$ and $Y \circ X$.

```
sage: X = sign_vector([-1,0,1,-1,0])
sage: X
(-0+-0)
sage: Y = sign_vector([0,1,0,1,0])
sage: Y
(0+0+0)

sage: X.compose(Y)
(-++-0)
sage: Y.compose(X)
(-+++0)
```

One can also use the operator & to compose sign vectors.

```
sage: X & Y
(-++-0)
sage: Y & X
(-+++0)
```

3.4 Separating elements

Two sign vectors are separated in a component if the corresponding entries have opposite signs.

Definition 3.12. Let $X, Y \in \{-, 0, +\}^E$. An element $e \in E$ *separates* X and Y if

$$X_e = -Y_e \neq 0.$$

We define the set of all elements that separate X and Y by

$$D(X, Y) := \{e \in E \mid X_e = -Y_e \neq 0\}.$$

Example 3.13. We take the set $E = \{1, 2, 3, 4, 5\}$ and consider the sign vectors

$$X = (\ 0\ -\ +\ 0\ +\ ) \quad \text{and} \quad Y = (+\ +\ +\ -\ -\ ).$$

We are looking for components with opposite signs. This applies to the second and the fifth component. Therefore, we obtain

$$D(X, Y) = \{2, 5\}.$$

There is a method implemented in the package that computes the set $D(X, Y)$.

Example 3.14. We define two sign vectors and compute the set of separating elements with Sage.

```
sage: X = sign_vector([0,-1,1,0,1])
sage: X
(0-+0+)
sage: Y = sign_vector([1,1,1,-1,-1])
```

```
sage: Y
(+++--)
sage: X.separating_elements(Y)
[1, 4]
```

In general, $X \circ Y \neq Y \circ X$ as we have seen in Example 3.10. In fact, composition is commutative if there are no separating elements.

Proposition 3.15. *Let $X, Y \in \{-, 0, +\}^E$. Then, $X \circ Y = Y \circ X$ if and only if $D(X, Y) = \emptyset$.*

Proof. Assume first that $X \circ Y = Y \circ X$. Let $e \in E$. If $X_e = 0$ or $Y_e = 0$, then $e \notin \mathcal{D}(X, Y)$. Next, assume that $X_e \neq 0$ and $Y_e \neq 0$. With

$$X \circ Y = Y \circ X,$$

we obtain

$$X_e = (X \circ Y)_e = (Y \circ X)_e = Y_e$$

and we conclude that $e \notin \mathcal{D}(X, Y)$ and thus $D(X, Y) = \emptyset$.

Conversely, let $D(X, Y) = \emptyset$ and let $e \in E$. If $X_e = 0$, then

$$X_e \circ Y_e = Y_e = Y_e \circ X_e.$$

For the other case, assume that $X_e \neq 0$. Since $D(X, Y) = \emptyset$, we have either $Y_e = 0$ or $Y_e = X_e$. Therefore,

$$X_e \circ Y_e = X_e = Y_e \circ X_e.$$

Thus, we conclude that $X \circ Y = Y \circ X$.

□

3.5 Conformal relation

We have already defined a partial order on sign vectors in Section 2.1. Now, we will extend this definition.

Definition 3.16. Let $X, Y \in \{-, 0, +\}^E$ be two sign vectors. We say X *is a face of* Y or X *conforms to* Y (denoted by $X \preceq Y$) if $X_e \neq 0$ implies $X_e = Y_e$. If $X \preceq Y$ and $X \neq Y$, we write $X \prec Y$.

We define further $0 \preceq +$ and $0 \preceq -$.

Example 3.17. Consider the sign vector

$$X = (\ 0\ +\ -\ 0\ \ 0\ ).$$

The sign vector X conforms a sign vector Y, that is, $X \preceq Y$, if Y is of the form

$$Y = (\ a\ +\ -\ b\ \ c\ ),$$

where a, b, c can be chosen arbitrarily from the set $\{-, 0, +\}$.

The conformal relation is also implemented in the package.

Example 3.18. We define three sign vectors X, Y and Z in Sage.

```
sage: X = sign_vector([-1,1,0,0,1])
sage: Y = sign_vector([-1,1,1,0,1])
sage: Z = sign_vector([-1,1,1,-1,1])
sage: X
(-+00+)
sage: Y
(-++0+)
sage: Z
(-++-+)
```

Then, we test the method `conforms`.

```
sage: X.conforms(Y)
True
sage: X.conforms(X)
True
sage: Y.conforms(Z)
True
sage: Z.conforms(Y)
False
sage: X.conforms(Z)
True
```

Therefore, we have

$$X \prec Y \prec Z.$$

The package also supports the usual operators <, <=, > and >=.

```
sage: X < X
False
sage: X <= Y
True
sage: Y > Z
False
sage: Z >= X
True
```

The following propositions follow immediately from the definition of composition.

Proposition 3.19. *Let $X, Y \in \{-, 0, +\}^E$. Then, $X \preceq X \circ Y$.*

Proof. Let $e \in E$. We need to show that $X_e \preceq (X \circ Y)_e$. If $X_e = 0$, then

$$0 = X_e \preceq Y_e = (X \circ Y)_e$$

and if $X_e \neq 0$, then

$$X_e = (X \circ Y)_e$$

and the result follows. □

Proposition 3.20. *Let $X, Y, Z \in \mathcal{S}$ with $Y \preceq Z$. Then, $X \circ Y \preceq X \circ Z$.*

Proof. Let $e \in E$ and let $X, Y, Z \in \mathcal{S}$ with $Y \preceq Z$. If $X_e = 0$, then

$$(X \circ Y)_e = Y_e \preceq Z_e = (X \circ Z)_e.$$

Conversely, if $X_e \neq 0$, then

$$(X \circ Y)_e = X_e = (X \circ Z)_e.$$

□

3.6 Orthogonality

Orthogonality for sign vectors is motivated by the following observation for real vectors. Assume that the standard scalar product of two real vectors $x, y \in \mathbb{R}^E$ is zero. Then, those vectors have either disjoint support or there exist $e, f \in E$ such that

$$x_e y_e > 0 \quad \text{and} \quad x_f y_f < 0.$$

Definition 3.21. Two sign vectors $X, Y \in \{-, 0, +\}^E$ are *orthogonal* if either $\underline{X} \cap \underline{Y} = \emptyset$ or if there are $e, f \in \underline{X} \cap \underline{Y}$ with $X_e = Y_e$ and $X_f = -Y_f$. In this case, we write $X * Y$.

Hence, if two real vectors are orthogonal, then also the corresponding sign vectors are orthogonal.

The package [sign_vectors] offers a method to check orthogonality between two sign vectors.

Example 3.22. First, we define several real vectors.

```
sage: x = vector([0,0,1])
sage: y = vector([1,-2,0])
sage: z = vector([2,1,2])
```

Next, we compute their standard scalar products.

```
sage: x.dot_product(y)
0
sage: y.dot_product(z)
0
sage: x.dot_product(z)
2
```

Hence, x is orthogonal to y and y is orthogonal to z, but x is not orthogonal to z. Now, we compute the corresponding sign vectors.

```
sage: X = sign_vector(x)
sage: X
(00+)
sage: Y = sign_vector(y)
sage: Y
(+-0)
sage: Z = sign_vector(z)
sage: Z
(+++)
```

We apply the corresponding method to check if those sign vectors are orthogonal.

```
sage: X.is_orthogonal_to(Y)
True
sage: Y.is_orthogonal_to(Z)
True
sage: X.is_orthogonal_to(Z)
False
```

Therefore, we obtain $X * Y$ and $Y * Z$, but x is not orthogonal to z.

If two real vectors are not orthogonal, then the corresponding sign vectors can still be orthogonal.

```
sage: u = vector([1,1,0])
sage: y.dot_product(u)
-1

sage: U = sign_vector(u)
sage: U
(++0)
sage: Y.is_orthogonal_to(U)
True
```

Thus, y is not orthogonal to u but still $Y * U$.

Analogously as for subsets of $\mathbb{R}^E$, we introduce the orthogonal complement of a set of sign vectors.

Definition 3.23. Let $\mathcal{S} \subseteq \{-, 0, +\}^E$. The *orthogonal complement* of $\mathcal{S}$ is the set

$$\mathcal{S}^\perp := \{X \in \{-, 0, +\}^E \mid X * Y \text{ for all } Y \in \mathcal{S}\}.$$

We can compute first the orthogonal complement of a subspace and then the corresponding sign vectors. The next result shows that we will obtain the same result, if we compute the orthogonal complement of the sign vectors corresponding to the subspace.

Proposition 3.24. *Let $\mathcal{V}$ be a subspace of $\mathbb{R}^E$. Then,* $\operatorname{sign}(\mathcal{V}^\perp) = \operatorname{sign}(\mathcal{V})^\perp$.

For a proof of Proposition 3.24 based on Minty's Lemma, see for example [MHR19].

3.7 Closure

We define the closure of a set of sign vectors as in [MHR19].

Definition 3.25. The *closure* of a subset $\mathcal{S} \subseteq \{-, 0, +\}^E$ is defined as the set

$$\overline{\mathcal{S}} := \{X \in \{-, 0, +\}^E \mid \text{there exists } Y \in \mathcal{S} \text{ with } X \preceq Y\}.$$

Now, we state a useful result about closure.

Theorem 3.26. *Let $\emptyset \neq \mathcal{S} \subseteq \{-, 0, +\}^E$. Then, $\overline{\mathcal{S}} \cap \mathcal{S}^* = \{0\}$.*

Proof. Let $X \in \overline{\mathcal{S}} \cap \mathcal{S}^*$. Since $X \in \overline{\mathcal{S}}$, there is $Y \in \mathcal{S}$ such that $X \preceq Y$. In addition, $X \in \mathcal{S}^*$ implies that for all $Z \in \mathcal{S}$, we have $X * Z$. In particular, we have $X * Y$.
There are two cases. If $X = \mathbf{0}$, we are done. Otherwise, since $X * Y$, there are $e, f \in \underline{X} \cap \underline{Y}$ such that $X_e = Y_e$ and $X_f = -Y_f$. But then $X \not\preceq Y$. Thus, we conclude that $X = \mathbf{0}$. □

We can use the package [sign_vectors] to compute the closure of a list of sign vectors.

```
sage: from sign_vectors import closure
```

Example 3.27. We define the sign vector X:

```
sage: X = sign_vector('0++-')
sage: X
(0++-)
```

To compute the closure of the set $\{X\}$, we apply the function `closure`:

```
sage: closure([X])
[(0000), (0+00), (00+0), (000-), (0++0), (0+0-), (00+-), (0++-)]
```

Example 3.28. Now, we consider a set consisting of two sign vectors:

```
sage: X = sign_vector('0++-')
sage: X
(0++-)
sage: Y = sign_vector('+---')
sage: Y
(+---)
```

We compute the closure:

```
sage: closure([X, Y])
[(0000), (+000), (0+00), (0-00), (00+0), (00-0), (000-), (+-00), (+0-0),
   (+00-), (0++0), (0+0-), (0--0), (0-0-), (00+-), (00--), (+--0),
   (+-0-), (+0--), (0++-), (0---), (+---)]
```

One can pass `separate=True` as optional argument to `closure`. That way, the computed closure will be split into lists. The sign vectors in each list have the same number of zero entries.

```
sage: cl = closure([X, Y], separate=True)
sage: cl
[[(0000)], [(+000), (0+00), (0-00), (00+0), (00-0), (000-)], [(+-00),
   (+0-0), (+00-), (0++0), (0+0-), (0--0), (0-0-), (00+-), (00--)],
   [(+--0), (+-0-), (+0--), (0++-), (0---)], [(+---)]]
```

Therefore, we have:

```
sage: cl[0]
[(0000)]
sage: cl[1]
[(+000), (0+00), (0-00), (00+0), (00-0), (000-)]
sage: cl[2]
[(+-00), (+0-0), (+00-), (0++0), (0+0-), (0--0), (0-0-), (00+-), (00--)]
sage: cl[3]
[(+--0), (+-0-), (+0--), (0++-), (0---)]
sage: cl[4]
[(+---)]
```

3.8 Positive and negative sign vectors

In Section 5, we will work with sign vectors that have no negative (or positive) component. Here, we will introduce a shortcut for a sign vector $X \in \{-, 0, +\}^E$ by writing $X \leq 0$ if $X_e \in \{-, 0\}$ for every $e \in E$. Similarly, we write $X \geq 0$ if $-X \leq 0$. Furthermore, we define $X < 0$ if $X \leq 0$ and $X \neq \mathbf{0}$ and analogously $X > 0$ if $X \geq 0$ and $X \neq \mathbf{0}$. In these cases, we call X negative or positive, respectively.
These operations are also implemented in the package [sign_vectors].

Example 3.29. We define a sign vector.

```
sage: X = sign_vector('++0')
sage: X
(++0)
```

Now, we check the operations from above.

```
sage: X > 0
True
sage: X <= 0
False
sage: zero_sign_vector(5) >= 0
True
sage: zero_sign_vector(5) < 0
False
```

4 Oriented matroids

An oriented matroid is a set of sign vectors, that satisfies certain axioms. These axioms relate oriented matroids closely to real subspaces. An oriented matroid contains the zero vector, negated sign vectors, and composition of sign vectors. There is a fourth property which states that we can find a sign vector with a zero component, if there are two sign vectors with opposite signs at the corresponding component.
A main reference of this chapter is [Fin01]. For further results, consult [BLS99] and [Zie95].

Definition 4.1. Let E be a finite set and let $\mathcal{F} \subseteq \{-, 0, +\}^E$ be a set of sign vectors. Then, the pair $\mathcal{M} := (E, \mathcal{F})$ is an *oriented matroid* if the following four *covector axioms* hold:

(F0) $\mathbf{0} \in \mathcal{F}$.

(F1) (symmetry)
If $X \in \mathcal{F}$, then $-X \in \mathcal{F}$.

(F2) (composition)
If $X, Y \in \mathcal{F}$, then $X \circ Y \in \mathcal{F}$

(F3) (covector elimination)
For all $X, Y \in \mathcal{F}$ and $e \in D(X, Y)$ there exists $Z \in \mathcal{F}$ with $Z_e = 0$ and for all $f \in E \setminus D(X, Y)$, we have $Z_f = (X \circ Y)_f$.

Here, we call each sign vector in $\mathcal{F}$ a *covector* or *face* of the oriented matroid $\mathcal{M}$.

One obtains an oriented matroid simply by taking a real subspace and applying the sign function to each vector of this subspace.

Proposition 4.2. *Let $\mathcal{V}$ be a subspace of $\mathbb{R}^E$. Then, the pair $\mathcal{M} := (E, \operatorname{sign}(\mathcal{V}))$ is an oriented matroid. In this case, we call $\mathcal{M}$ the* oriented matroid corresponding to the subspace $\mathcal{V}$.

Given a real subspace, it is not difficult to see that the corresponding oriented matroid will fulfil (F0), (F1) and (F2). Now, we show an example to illustrate that (F3) is satisfied, too.

Example 4.3. Let $\mathcal{V}$ be a real subspace and let $x, y \in \mathcal{V}$ be the vectors

$$x = (2, 1, 0) \quad \text{and} \quad y = (-1, 2, 1).$$

The first components have opposite signs. To eliminate this component, we multiply y by 2 and add it to x. Therefore, we obtain

$$x + 2y = (2, 1, 0) + 2\ (-1, 2, 1) = (0, 5, 2) =: z \in \mathcal{V}.$$

The corresponding sign vectors are

$$X = (+\ +\ 0\ ), \quad Y = (-\ +\ +) \quad \text{and} \quad Z = (\ 0\ +\ +).$$

Therefore, Z is the covector we obtain from X and Y by covector elimination (F3).

Note that there are oriented matroids that cannot be obtained from a real subspace.

Definition 4.4. An oriented matroid is called *realizable* if it can be obtained by applying sign to each vector of a real subspace.

For some results, we cannot apply covector elimination (F3) directly. There is an equivalent axiom called conformal elimination.

(F3c) (conformal elimination)
For all $X, Y \in \mathcal{F}$ and $\emptyset \neq S \subseteq D(X,Y)$ there is $e \in S$ and $Z \in \mathcal{F}$ such that $Z_e = 0$ and $Z_S \preceq X_S$ and $Z_f = (X \circ Y)_f$ for all $f \in E \setminus D(X,Y)$.

One can apply conformal elimination to real vectors as well.

Example 4.5. Let $\mathcal{V}$ be a subspace of $\mathbb{R}^E$ with $E = \{1,2,3,4,5\}$ and let $x, y \in \mathcal{V}$ be defined as

$$x = (1,6,2,0,-1) \quad \text{and} \quad y = (0,-2,-1,-2,-2).$$

The corresponding sign vectors are

$$X = (+\,+\,+\;0\;-) \quad \text{and} \quad Y = (\;0\;-\,-\,-\,-).$$

The second and third component have opposite signs. We choose $S := D(X,Y) = \{2,3\}$. By adding a positive multiple of y to x, we can either eliminate the second or the third component. Therefore, we obtain two vectors generated by x and y, that is,

$$\begin{aligned} x + 3y &= (1,6,2,0,-1) + 3\,(0,-2,-1,-2,-2) = (1,0,-1,-6,-7) =: v \in \mathcal{V}, \\ x + 2y &= (1,6,2,0,-1) + 2\,(0,-2,-1,-2,-2) = (1,2,0,-4,-5) =: w \in \mathcal{V}. \end{aligned}$$

The corresponding sign vectors are

$$V = (+\;0\;-\,-\,-) \quad \text{and} \quad W = (+\,+\;0\;-\,-).$$

If we restrict those sign vectors to S, we obtain

$$\begin{aligned} V_S &= (\;0\;-\;) \not\preceq (+\,+) = X_S, \\ W_S &= (+\;0\;) \preceq (+\,+) = X_S. \end{aligned}$$

Since W satisfies $W_S \preceq X_S$, we will proceed with the vector w. To show the last property of (F3c), we define $R := \{1,2,3,4,5\} \setminus D(X,Y) = \{1,4,5\}$. Then,

$$W_R = (+\,-\,-) = (+\;0\;-) \circ (\;0\;-\,-) = X_R \circ Y_R.$$

Hence, there is indeed $e \in S$, that is, $e = 3$, such that the properties in (F3c) are satisfied. The corresponding covector is W.

The procedure we applied in Example 4.5 to obtain the real vector w can be formalized. An implementation of the corresponding algorithm is offered by the package [`elementary_vectors`].

```
sage: from elementary_vectors import conformal_elimination
```

Example 4.6. We define two real vectors x and y.

```
sage: x = vector([2,-1,0,3,2,-4,0,2])
sage: y = vector([-1,-2,0,4,-3,3,1,-2])
```

Then, we apply the function `conformal_elimination` to x and y to compute a new vector z.

```
sage: z = conformal_elimination(x,y)
sage: z
```

```
(4/3, -7/3, 0, 17/3, 0, -2, 2/3, 2/3)
```

One can see from the result that the fifth component, that is, the component with index 4, has been eliminated. Next, we calculate the corresponding sign vectors.

```
sage: X = sign_vector(x)
sage: Y = sign_vector(y)
sage: Z = sign_vector(z)
sage: X
(+-0++-0+)
sage: Y
(--0+-++-)
sage: Z
(+-0+0-++)
```

In order to check (F3c) for these sign vectors, we compute the set of separating elements $D(X, Y)$.

```
sage: D = X.separating_elements(Y)
sage: D
[0, 4, 5, 7]
```

We restrict X, Y and Z to $D(X, Y)$.

```
sage: X_D = sign_vector(X.list_from_positions(D))
sage: X_D
(++-+)
sage: Y_D = sign_vector(Y.list_from_positions(D))
sage: Y_D
(--+-)
sage: Z_D = sign_vector(Z.list_from_positions(D))
sage: Z_D
(+0-+)
sage: Z_D <= X_D
True
```

Therefore,

$$Z_D \preceq X_D.$$

Now, we define $R := E \setminus D(X, Y) = \{1, 2, 3, 6\}$.

```
sage: R = [1,2,3,6]
```

We restrict our sign vectors to R.

```
sage: X_R = sign_vector(X.list_from_positions(R))
sage: X_R
(-0+0)
sage: Y_R = sign_vector(Y.list_from_positions(R))
sage: Y_R
(-0++)
```

```
sage: Z_R = sign_vector(Z.list_from_positions(R))
sage: Z_R
(-0++)
sage: Z_R == X_R & Y_R
True
```

Hence,

$$Z_R = X_R \circ Y_R.$$

Therefore, (F3c) holds for the sign vectors corresponding to x, y and z.

As we have mentioned before, covector elimination (F3) is equivalent to conformal elimination (F3c). There is another equivalent axiom called weak elimination. For further details on these axioms and a proof of the corresponding equivalence, see [Fin01]. Since we will not need weak elimination, we will only show the equivalence between covector elimination (F3) and conformal elimination (F3c).

Proposition 4.7. *Let $\mathcal{F} \subseteq \{-, 0, +\}^E$ such that (F0), (F1) and (F2) hold. Then, (F3) is equivalent to (F3c).*

Proof. Assume that (F3) holds. Let $X, Y \in \mathcal{F}$ and let $\emptyset \neq S \subseteq D(X, Y)$. We proceed by induction on $|S|$.
For the initial step, assume that $|S| = 1$. By (F3), there exists $Z \in \mathcal{F}$ with $Z_e = 0$ and $Z_f = (X \circ Y)_f$ for all $f \in E \setminus D(X, Y)$. Furthermore, $Z_S \preceq X_S$ since $S = \{e\}$.
To show the inductive step, assume that $|S| > 1$ and assume that (F3c) is satisfied for all

$$\emptyset \neq T \subseteq D(X, Y)$$

with $|T| < |S|$.
Next, choose $e \in S$ and define $T := S \setminus \{e\}$. By the induction hypothesis, there is $Z' \in \mathcal{F}$ with $Z'_T \preceq X_T$ and $Z'_f = (X \circ Y)_f$ for all $f \in E \setminus D(X, Y)$.
We consider two cases. First, let $e \notin D(X, Z')$. Since $e \in S \subseteq D(X, Y)$, we obtain $X_e \neq 0$. Hence, with $e \notin D(X, Z')$, we have $Z'_e \preceq X_e$. Therefore,

$$Z'_S \preceq X_S$$

and (F3c) follows.
For the other case, assume that $e \in D(X, Z')$. We apply (F3) to X, Z' and e. Therefore, there is $Z \in \mathcal{F}$ such that $Z_e = 0$ and $Z_f = (X \circ Z')_f$ for all $f \in E \setminus D(X, Z')$.
It remains to show that $Z_S \preceq X_S$ holds. First, $Z_e = 0 \preceq X_e$. Since $Z'_T \preceq X_T$, we obtain $T \subseteq E \setminus D(X, Z')$. Furthermore, $T \subseteq D(X, Y) \subseteq \underline{X}$. Hence,

$$Z_T = (X \circ Z')_T = X_T.$$

In addition, let $f \in E \setminus D(X, Y)$. Then, we obtain $Z'_f = (X \circ Y)_f$ and therefore $f \in E \setminus D(X, Z')$ and consequently

$$Z_f = (X \circ Z')_f = (X \circ Y)_f.$$

To show the other direction, assume that (F3c) holds. Let $X, Y \in \mathcal{F}$ and $e \in D(X, Y)$. We define $S := \{e\}$ and apply (F3c) to X, Y and S. Hence, there exists $Z \in \mathcal{F}$ with $Z_e = 0$ and $Z_f = (X \circ Y)_f$ for all $f \in E \setminus D(X, Y)$. Thus, (F3) holds. □

In the remainder of this thesis, let $\mathcal{M} = (E, \mathcal{F})$ be an oriented matroid. Therefore, $\mathcal{F}$ is the corresponding set of covectors on E.

Now, we use the conformal relation $\preceq$ to introduce a lattice on an oriented matroid $\mathcal{M}$. The bottom element of this lattice will be the zero sign vector $\mathbf{0}$. Except for the trivial case, there is no top element for an oriented matroid. Therefore, we add a new element $\mathbf{1}$ to $\mathcal{F}$ which serves as the top element. The resulting lattice will be called the *big face lattice*. Further details on face lattices are available in [Fin01] and [Zie95]. For basic definitions on lattices, inspect Appendix B or [Grä78].

Lemma 4.8. *The partially ordered set*

$$\tilde{\mathcal{F}}(\mathcal{M}) := (\mathcal{F}, \preceq)$$

is a lattice. Here, $\tilde{\mathcal{F}} := \mathcal{F} \cup \{\mathbf{1}\}$ *and* $\preceq$ *is the conformal relation extended by* $X \preceq \mathbf{1}$ *for all* $X \in \tilde{\mathcal{F}}$.

Proof. Let $X, Y \in \tilde{\mathcal{F}}$. First, we show that $\sup(X, Y) \in \tilde{\mathcal{F}}$. If $X \preceq Y$ or $Y \preceq X$, then $\sup(X, Y) = Y \in \tilde{\mathcal{F}}$ or $\sup(X, Y) = X \in \tilde{\mathcal{F}}$, respectively.
Now, assume that neither $X \preceq Y$ nor $Y \preceq X$. We consider two cases. If $D(X, Y) = \emptyset$ then by Proposition 3.15 and (F2), we obtain

$$X \circ Y = Y \circ X \in \tilde{\mathcal{F}}.$$

Next, we apply Proposition 3.19 and obtain

$$X, Y \preceq X \circ Y.$$

By the definition of composition, we obtain

$$\sup(X, Y) = X \circ Y \in \tilde{\mathcal{F}}.$$

On the other hand, if $D(X, Y) \neq \emptyset$ then there is no $Z \in \mathcal{F}$ with $X, Y \preceq Z$. Thus, we set $\sup(X, Y) = \mathbf{1} \in \tilde{\mathcal{F}}$.

It remains to show that $\inf(X, Y) \in \tilde{\mathcal{F}}$. If $X \preceq Y$ or $Y \preceq X$, then $\inf(X, Y) = X \in \tilde{\mathcal{F}}$ or $\inf(X, Y) = Y \in \tilde{\mathcal{F}}$, respectively.
Otherwise X and Y are not comparable. Next, we consider the set

$$L := \{Z \in \mathcal{F} \mid Z \preceq X \text{ and } Z \preceq Y\}.$$

Since $L \subseteq \mathcal{F}$ and $\mathcal{F}$ is finite, there exists $l \in \mathbb{N}$ and $Z^1, \dots, Z^l \in \mathcal{F}$ such that

$$L = \{Z^1, \dots, Z^l\}.$$

Then, we can set $\inf(X, Y) = Z^1 \circ \cdots \circ Z^l \in \mathcal{F}$. We show that the infimum is uniquely defined, that is, the order of the Z^i does not matter in this composition. Let $e \in E$. Assume there are $i, j \in \{1, \dots, l\}$ such that

$$Z^i_e = + \quad \text{and} \quad Z^j_e = -.$$

This cannot be the case since we would obtain $+ \preceq X_e$ and $- \preceq X_e$ because $Z^i, Z^j \in L$. Therefore, either $Z^i_e \in \{-, 0\}$ for every $i \in \{1, \dots, l\}$ or $Z^i_e \in \{0, +\}$ for every $i \in \{1, \dots, l\}$. We conclude that the infimum $\inf(X, Y)$ is uniquely determined. □

4.1 Cocircuits

Cocircuits are minimal non-zero elements with respect to $\preceq$. In other words, cocircuits cover the zero vector in the big face lattice.

Definition 4.9. A covector $V \in \mathcal{F} \setminus \{0\}$ is called a *cocircuit* of $\mathcal{M}$ if for every $X \in \mathcal{F} \setminus \{0\}$, we have that $X \preceq V$ implies $X = V$. By

$$\mathcal{D} := \min(\mathcal{F} \setminus \{\mathbf{0}\}) = \{V \in \mathcal{F} \mid V \text{ is a cocircuit}\},$$

we denote the set of *cocircuits* of $\mathcal{M}$.

Elementary vectors of a real subspace play a similar role as cocircuits. Cocircuits are the non-zero covectors with minimal support.

Lemma 4.10. *Cocircuits are support-minimal.*

Proof. Let $V \in \mathcal{F}$ be a cocircuit and let $X \in \mathcal{F} \setminus \{\mathbf{0}\}$ with $\underline{X} \subseteq \underline{V}$. Assume that neither $X \preceq V$ nor $-X \preceq V$. Otherwise, V is no cocircuit. Therefore, $D := D(V, X) \neq \emptyset$ and $D(V, -X) \neq \emptyset$. We apply conformal elimination (F3c) to V, X and D. Thus, there are $e \in S$ and $Z \in \mathcal{F}$ with

$$Z_e = 0, \quad Z_D \preceq V_D \quad \text{and} \quad Z_f = (V \circ X)_f$$

for all $f \in E \setminus D(V, X)$. Consequently, we have $Z \prec V$. It remains to show that $Z \neq \mathbf{0}$. Since $D(V, -X) \neq \emptyset$, there is $g \in E \setminus D(V, X)$ with $V_g = X_g \neq 0$. Thus, $Z_g = (V \circ X)_g \neq 0$ and therefore

$$\mathbf{0} \neq Z \prec V,$$

which is a contradiction to V being a cocircuit. Therefore, we conclude that V is support-minimal. □

Therefore, one can obtain cocircuits by applying the sign function to the elementary vector.
The package [sign_vectors] offers many functions to compute with oriented matroids. We load the corresponding module to have access to these functions.

```
sage: from sign_vectors.oriented_matroids import *
```

Example 4.11. We consider the following matrix.

```
sage: A = matrix([[1,0,2],[0,1,-1]])
sage: A
[ 1  0  2]
[ 0  1 -1]
```

Next, we compute a list of cocircuits that correspond to the rows of A.

```
sage: ccA = cocircuits_from_matrix(A)
sage: ccA
[(--0), (-0-), (0-+), (++0), (+0+), (0+-)]
```

Since the matrix A is in echelon form, the row vectors are already support minimal. Therefore, the corresponding sign vectors

$$(+\ 0\ +) \quad \text{and} \quad (\ 0\ +\ -\ )$$

are already cocircuits. We can make a linear combination of the rows such that the third component vanishes. By applying the sign function to this linear combination, we find another cocircuit, that is

$$(+\,+\,0\,).$$

The remaining cocircuits can by determined by reversing the signs of the previously computed cocircuits.

Example 4.12. Now, we consider another matrix.

```
sage: B = matrix([[1,0,0,0],[0,1,0,2],[0,0,1,-1]])
sage: B
[ 1  0  0  0]
[ 0  1  0  2]
[ 0  0  1 -1]
```

We compute the cocircuits that correspond to B.

```
sage: ccB = cocircuits_from_matrix(B)
sage: ccB
[(+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-)]
```

The following results on cocircuits are also available in [Fin01]. An important property of cocircuits is that every non-zero covector has a cocircuit such that the cocircuit conforms to that covector. In addition, we can fix a non-zero entry of the covector such that the corresponding cocircuit has the same non-zero entry.

Lemma 4.13. *Let $X \in \mathcal{F}$ and $e \in \underline{X}$. Then, there exists a cocircuit $V \in \mathcal{D}$, with $V \preceq X$ and $V_e = X_e$.*

Proof. We define

$$\hat{\mathcal{F}} := \{Y \in \mathcal{F} \mid Y \preceq X \text{ and } Y_e = X_e\}.$$

Since $X \in \hat{\mathcal{F}}$, we obtain $\hat{\mathcal{F}} \neq \emptyset$. We show that there is a cocircuit in $\hat{\mathcal{F}}$. Let $V \in \hat{\mathcal{F}}$ such that there is no $Y \in \hat{\mathcal{F}}$ with $Y \prec V$, that is, V is a minimal element in $\hat{\mathcal{F}}$ with respect to $\preceq$.
Seeking a contradiction, we assume that V is not a cocircuit. Next, assume there is $W \in \mathcal{F} \setminus \{\mathbf{0}\}$ with $W \prec V \preceq X$. Since V is minimal in $\hat{\mathcal{F}}$, we have $W_e = 0$.
Note that $S := D(V, -W) = \underline{W} \neq \emptyset$. Thus, we can apply conformal elimination (F3c) to $V, -W$ and S. Hence, there are $f \in S$ and $Z \in \mathcal{F}$ such that

$$Z_f = 0, \quad Z_S \preceq V_S \quad \text{and} \quad Z_g = (V \circ -W)_g \text{ for all } g \in E \setminus S.$$

Assume there exists $h \in E \setminus S$ such that $0 \neq Z_h \neq V_h$. Then, $V_h = 0$ and $Z_h = -W_h \neq 0$. Thus, $V_h \prec W_h$ which is a contradiction to $W \prec V$. Therefore, $Z \preceq V$.
With $Z_f = 0 \neq V_f$, we obtain $Z \neq V$. Furthermore, $Z_e = X_e$ since $W_e = 0$ and $Z_e = (V \circ (-W))_e = V_e = X_e \neq 0$. Hence, $Z \prec V$ and $Z \in \hat{\mathcal{F}}$. This contradicts the minimality of V in $\hat{\mathcal{F}}$. Thus, there is a cocircuit in $\hat{\mathcal{F}}$. □

The next theorem states that every covector can be written as a conformal composition of cocircuits.

Proposition 4.14. *We can represent every covector $X \in \mathcal{F} \setminus \{\mathbf{0}\}$ as*

$$X = V^1 \circ V^2 \circ \cdots \circ V^l,$$

where $V^i \in \mathcal{D}$ and $V^i \preceq X$ for all $i \in \{1, \ldots, l\}$. There is a conformal decomposition of X with $l \leq |\underline{X}|$.

Proof. By Lemma 4.13, there is for every $e \in \underline{X}$ a cocircuit $V^e \in \mathcal{D}$ such that

$$V^e \preceq X \quad \text{and} \quad V^e_e = X_e.$$

We can set

$$\{V^1, \ldots, V^l\} := \{V^e \mid e \in \underline{X}\}.$$

□

According to Theorem 2.8, every vector of a real subspace can be written as a conformal sum of elementary vectors. Similarly, we can use a set of cocircuits, to determine every covector of an oriented matroid.

Corollary 4.15. *The set $\mathcal{D}$ of cocircuits of $\mathcal{M}$ determines the set of covectors by*

$$\mathcal{F} = \{X \mid X = V^1 \circ V^2 \circ \cdots \circ V^l \text{ for } V^i \in \mathcal{D} \text{ such that } V^i \preceq X, l \geq 1\} \cup \{0\}.$$

Proof. By Proposition 4.14, every covector $X \in \mathcal{F} \setminus \{\mathbf{0}\}$ can be obtained by composing cocircuits V^i with $V^i \preceq X$. □

Corollary 4.15 leads us to an algorithm that is described in [Fin01]. By applying composition to cocircuits in an organized way, we can determine every covector. An implementation of this algorithm is available in the package [`sign_vectors`],

Example 4.16. We continue with the matrix

$$A = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \end{pmatrix}$$

from Example 4.11. The cocircuits corresponding to the rows of A are:

```
sage: ccA
[(--0), (-0-), (0-+), (++0), (+0+), (0+-)]
```

Next, we apply `covectors_from_cocircuits` to the list of cocircuits.

```
sage: cvA = covectors_from_cocircuits(ccA)
sage: cvA
[(000), (--0), (-0-), (0-+), (++0), (+0+), (0+-), (---), (-+-), (++-),
   (+++), (--+), (+-+)]
```

We plot the corresponding Hasse diagram.

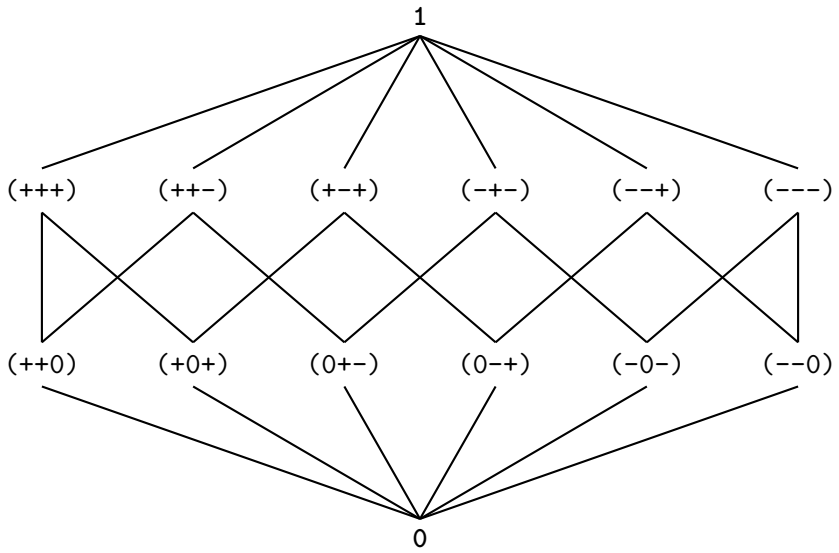


Figure 4.1: Big face lattice corresponding to A

Example 4.17. Now, we consider the matrix

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

from Example 4.12. The list of cocircuits is:

```
sage: ccB
[(+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-)]
```

We apply `covectors_from_cocircuits` and obtain:

```
sage: cvB = covectors_from_cocircuits(ccB)
sage: cvB
[(0000), (+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-),
   (+0+-), (0---), (0-+-), (-0+-), (0++-), (0+++), (++++), (0--+),
   (0+-+), (-+++), (---+), (--+-), (-+-+), (-++-), (+++-), (----),
   (+---), (+--+), (++-+), (+-+-), (-0-+), (+0-+), (++0+), (-+0+),
   (--0-), (+-0-), (+++0), (-++0), (---0), (+--0)]
```

Now, we are interested in the top elements of the posets we could see in Example 4.11 and Example 4.12. We will call these elements topes.

4.2 Topes

In contrast to cocircuits, topes are the covectors which are maximal with respect to $\preceq$. In the big face lattice $\tilde{\mathcal{F}}$, topes are the elements that are covered by the element $\mathbf{1} \in \tilde{\mathcal{F}}$.

Definition 4.18. A covector $T \in \mathcal{F} \setminus \{0\}$ is called a *tope* of $\mathcal{M}$ if for every $X \in \mathcal{F} \setminus \{0\}$, we have that $T \preceq X$ implies $X = T$. By

$$\mathcal{T} := \max(\mathcal{F}) = \{T \in \mathcal{F} \mid T \text{ is a tope}\},$$

we denote the set of *topes* in $\mathcal{M}$.

If an entry is zero in each covector, we call this entry a loop.

Definition 4.19. Let $\mathcal{S} \subseteq \{-, 0, +\}^E$ and let $e \in E$. If $e \in X^0$ for all $X \in \mathcal{S}$, we call e a *loop* of $\mathcal{S}$.

The package [sign_vectors] offers a function to compute the loops of a list of sign vectors.

```
sage: from sign_vectors.utility import loops
```

Example 4.20. First, we define some sign vectors.

```
sage: X = sign_vector('00+0-+0')
sage: Y = sign_vector('0-00+0+')
sage: Z = sign_vector('0-+0-0+')
sage: X
(00+0-+0)
sage: Y
(0-00+0+)
sage: Z
(0-+0-0+)
```

We compute the loops of $\{X, Y, Z\}$.

```
sage: loops([X,Y,Z])
[0, 3]
```

If a tope has a zero entry, then every covector of the corresponding oriented matroid must also have this zero entry. Otherwise, we could apply composition and construct a larger covector.

Lemma 4.21. *A covector $X \in \mathcal{F}$ is a tope if and only if X^0 is the set of loops of $\mathcal{F}$.*

Proof. Let $X \in \mathcal{F}$ be a tope. We define $E^0 \subseteq E$ as the set of all loops of $\mathcal{F}$. Now, let $e \in E^0$. Since e is a loop, we conclude that $e \in X^0$. Hence, $E^0 \subseteq X^0$.
Next, let $e \in X^0$. Then, $X_e = 0$. Assume that $e \notin E^0$. Then, there is $Y \in \mathcal{F}$ such that $Y_e \neq 0$. Therefore, $X \prec X \circ Y$. This is a contradiction to X being a tope. Therefore, we obtain $X^0 \subseteq E^0$ and conclude that $X^0 = E^0$.

On the other hand, let $X \in \mathcal{F}$ not be a tope. Then, there is $Y \in \mathcal{F}$ such that $X \prec Y$. Thus, there is $e \in X^0$ such that $Y_e \neq 0$. Therefore, e is not a loop. □

As we have seen in Corollary 4.15, we can determine all covectors of an oriented matroid by the cocircuits. There is a similar result for topes.

Proposition 4.22. *Let $\mathcal{T}$ be the set of topes of the oriented matroid $\mathcal{M} = (E, \mathcal{F})$. Then, the set of covectors is given by*

$$\mathcal{F} = \{X \in \{-, 0, +\}^E \mid X \circ T \in \mathcal{T} \text{ for all } T \in \mathcal{T}\}.$$

Proof. Let $X \in \mathcal{F}$ and $T \in \mathcal{T}$. By (F2), $Y := X \circ T \in \mathcal{F}$. Furthermore, $Y^0 = T^0$. With Lemma 4.21, we obtain $Y \in \mathcal{T}$.
Now, let $X \in \{-, 0, +\}^E$ such that for all $T \in \mathcal{T}$, we have $X \circ T \in \mathcal{T}$. We show that $X \in \mathcal{F}$. For this purpose, we will prove that $X \circ Y \in \mathcal{F}$ for all $Y \in \mathcal{F}$.

We proceed by induction on $|(X \circ Y)^0|$. For the initial step, consider $Y \in \mathcal{F}$ such that $|(X \circ Y)^0|$ is minimal. Then, let $Z \in \mathcal{T}$ with $Y \preceq Z$. By Lemma 3.20, we obtain

$$X \circ Y \preceq X \circ Z$$

and therefore

$$|(X \circ Y)^0| \geq |(X \circ Z)^0|.$$

Since $|(X \circ Y)^0|$ is minimal, we have

$$|(X \circ Y)^0| = |(X \circ Z)^0|$$

and hence

$$X \circ Y = X \circ Z \in \mathcal{F}.$$

For the inductive step, consider $Y \in \mathcal{F}$ such that $|(X \circ Y)^0|$ is not minimal. Furthermore, assume that $X \circ Z \in \mathcal{F}$ for all $Z \in \mathcal{F}$ with $|(X \circ Y)^0| > |(X \circ Z)^0|$. Since $|(X \circ Y)^0|$ is not minimal, there exists $Z \in \mathcal{F}$ with

$$|(X \circ Y)^0| > |(X \circ Z)^0| \quad \text{and} \quad X \circ Y \preceq X \circ Z.$$

Therefore,

$$X \circ Y \prec X \circ Z \in \mathcal{F}.$$

Then, assume that there is no $Z' \in \mathcal{F}$ with

$$X \circ Y \prec X \circ Z' \prec X \circ Z. \tag{4.2}$$

Otherwise, we could take Z' instead of Z. Since $Y, Z \in \mathcal{F}$, we obtain $Y \circ Z \in \mathcal{F}$. Then, observe that

$$|(X \circ Y)^0| > |(X \circ Z)^0| \geq |(X \circ Y \circ Z)^0|.$$

Thus, we have

$$\begin{aligned} W^+ &:= X \circ Y \circ Z \in \mathcal{F} \\ W^- &:= X \circ Y \circ (-Z) \in \mathcal{F}. \end{aligned}$$

Since

$$X \circ Y \prec X \circ Y \circ Z = W^+,$$

we obtain

$$X \circ Y \prec X \circ Y \circ (-Z) = W^-$$

and therefore

$$D := D(W^+, W^-) \neq \emptyset.$$

We apply conformal elimination (F3c) to W^+, W^- and D and find $e \in D$ and $W \in \mathcal{F}$ with

$$W_e = 0, \quad W_D \prec W_D^+ \quad \text{and} \quad W_f = (W^+ \circ W^-)_f = W_f^+$$

for all $f \in E \setminus D$. Thus, $W \prec W^+$. Next, let $S := \underline{X}$. Then,

$$\underline{X} = X_S = W_S^+ = W_S^- = W_S$$

by definition of W^+, W^- and W. Thus, we obtain

$$X \circ W = W.$$

Furthermore,

$$X \circ Y \preceq X \circ Z$$

implies

$$W^+ = X \circ Y \circ Z = X \circ Z.$$

Thus, we conclude that

$$X \circ W = W \prec W^+ = X \circ Z.$$

In addition, we have

$$X \circ Y \preceq X \circ W.$$

With the assumption (4.2), we finally obtain

$$X \circ Y = X \circ W = W \in \mathcal{F}.$$

□

Applying composition to the cocircuits in a structured way leads to an algorithm which we can use to compute the topes. The algorithm is described in [Fin01] and an implementation is offered by the package [`sign_vectors`].

Example 4.23. We consider again the matrix

```
sage: A
[ 1  0  2]
[ 0  1 -1]
```

from Example 4.11. We have already computed the cocircuits defined by the rows of A.

```
sage: ccA
[(--0), (-0-), (0-+), (++0), (+0+), (0+-)]
```

Now, we use the Sage package to compute the topes by using the cocircuits.

```
sage: tA = topes_from_cocircuits(ccA)
sage: tA
[(---), (-+-), (++-), (+++), (--+), (+-+)]
```

These are precisely the elements on top of Figure 4.1.

Example 4.24. Next, we consider the matrix

```
sage: B
[ 1  0  0  0]
[ 0  1  0  2]
[ 0  0  1 -1]
```

from Example 4.12. We will use the cocircuits to compute the topes.

```
sage: ccB
[(+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-)]
```

Hence, we obtain the following topes:

```
sage: tB = topes_from_cocircuits(ccB)
sage: tB
[(++++), (-+++), (++-+), (-+-+), (+--+), (---+), (+++-), (-++-), (----),
   (--+-), (+---), (+-+-)]
```

4.3 Face enumeration

In Section 4.1, we found a way to compute all covectors using the cocircuits. Now, we will discuss a different approach by using the tope set.

4.3.1 Parallelness

First, we define the concept of parallelness associated with sign vectors. Given is a set of sign vectors on E. Two elements of E are parallel if for each sign vector either the corresponding components are equal or the signs differ by $-$.

Definition 4.25. Let $\mathcal{S}$ be a set of sign vectors on E. We call $e, f \in E$ *parallel elements* if there exists $\Lambda \in \{-, +\}$ such that for all $X \in \mathcal{S}$, we have $X_e = \Lambda X_f$.

We will show that parallelness is an equivalence relation. Therefore, we can partition the set E into *parallel classes*.

Lemma 4.26. *Parallelness is an equivalence relation.*

Proof. Let $\mathcal{S}$ be a set of sign vectors on E and let $e \in E$. Parallelness is reflexive since $X_e = X_e$ for all $X \in \mathcal{S}$.

Next, let $e, f \in E$ such that e, f are parallel elements. Hence, there is $\Lambda \in \{-, +\}$ such that

$$X_e = \Lambda X_f$$

for all $X \in \mathcal{S}$. Consequently,

$$X_f = \Lambda X_e$$

for all $X \in \mathcal{S}$ and therefore, we obtain symmetry.

Finally, let $e, f, g \in E$ with e being parallel to f and f being parallel to g. Hence, there are $\Lambda, \Theta \in \{-, +\}$ such that

$$X_e = \Lambda X_f \quad \text{and} \quad X_f = \Theta X_g$$

for all $X \in \mathcal{S}$. Therefore, $X_e = \Lambda\Theta X_g$ and consequently e is parallel to g. □

If a set of sign vectors contains a sign vector that has a zero and a non-zero component, then those two components cannot be parallel. This is Lemma 0.7.5 from [Fin01].

Lemma 4.27. *Two elements $e, f \in E$ are parallel elements of $\mathcal{F}$ if and only if there exists no covector $X \in \mathcal{F}$ such that exactly one of X_e and X_f is equal to 0.*

Proof. Let $X \in \mathcal{F}$ and assume that there are $e, f \in E$ with $X_e = 0$ and $X_f \neq 0$. By definition of parallelness, e and f are not parallel.

For the other direction, consider that there are $e, f \in E$ that are not parallel. Then, there are two cases. If there exists $X \in \mathcal{F}$ with exactly one of X_e and X_f equal to 0, we are done. Now, assume there exist $X, Y \in \mathcal{F}$ with

$$X_e = X_f \neq 0 \quad \text{and} \quad Y_e = -Y_f \neq 0.$$

By possibly swapping e and f, we have $X_e = -Y_e \neq 0$. Therefore, $e \in D(X, Y)$. Furthermore, we have $X_f = Y_f$ and thus $f \in E \setminus D(X, Y)$. We apply covector elimination (F3) to X, Y and e. Then, there exists $Z \in \mathcal{F}$ with $Z_e = 0$ and $Z_f = (X \circ Y)_f = X_f \neq 0$ which concludes this direction. □

There is also a utility function in the package [`sign_vectors`] that we can use to find parallel classes of a given list of sign vectors.

```
sage: from sign_vectors.utility import parallel_classes
```

Example 4.28. We define two sign vectors and determine their parallel classes.

```
sage: x = sign_vector([-1,1,1,0,1,0])
sage: x
(-++0+0)
sage: y = sign_vector([1,-1,1,0,1,1])
sage: y
(+-+0++)
sage: parallel_classes([x,y])
[[0, 1], [2, 4], [3], [5]]
```

Consequently, there are four parallel classes.

In terms of real subspaces, parallelness of components can be defined as follows:

Definition 4.29. Let $\mathcal{A}$ be a subset of $\mathbb{R}^E$. We call $e, f \in E$ *parallel elements* if there is $\lambda \in \mathbb{R} \setminus \{0\}$ with $x_e = \lambda x_f$ for all $x \in \mathcal{A}$.

Example 4.30. Consider the two vectors

$$(6, 3, 0) \quad \text{and} \quad (4, 2, -1).$$

Those vectors satisfy $x_1 = 2x_2$. Therefore, 1 and 2 are parallel.

Example 4.31. Consider now, the vectors

$$(1, -2, 1, 1) \quad \text{and} \quad (-2, 4, 0, 0).$$

Here, we have two parallel classes, that is, $S := \{1, 2\}$ and $T := \{3, 4\}$.

The utility function `parallel_classes` also works for a list of real vectors.

Example 4.32. We consider three real vectors and compute their parallel classes.

```
sage: x = vector([-1,1,2,5,1,0])
sage: y = vector([1,1,-2,1,1,0])
sage: z = vector([0,-3,0,-1,-3,1])
sage: parallel_classes([x,y,z])
[[0, 2], [1, 4], [3], [5]]
```

Therefore, we have four parallel classes.

One might think that the corresponding sign vectors have the same parallel classes. In general, this does not hold as we see in the following steps.

```
sage: X = sign_vector(x)
sage: X
(-++++0)
sage: Y = sign_vector(y)
sage: Y
(++-++0)
sage: Z = sign_vector(z)
sage: Z
(0-0--+)
sage: parallel_classes([X,Y,Z])
[[0, 2], [1, 3, 4], [5]]
```

There are only three parallel classes for the corresponding set of sign vectors.

On the contrary, the parallel classes of a real subspace are the same as the parallel classes of the corresponding oriented matroid.

Proposition 4.33. *Let $\mathcal{V}$ be a subspace of $\mathbb{R}^E$. Then, the parallel classes of $\mathcal{V}$ are the same as the parallel classes of* $\operatorname{sign}(\mathcal{V})$.

Proof. Let $e, f \in E$ be parallel in $\mathcal{V}$. Hence, there is $\lambda \in \mathbb{R}$ such that for all $x \in \mathcal{V}$, we have $x_e = \lambda x_f$. Now, let $X \in \mathcal{F}$ and let $x \in \mathcal{V}$ be the corresponding real vector. Then, we obtain

$$X_e = \operatorname{sign}(x)_e = \operatorname{sign}(x_e) = \operatorname{sign}(\lambda x_f) = \operatorname{sign}(\lambda)\operatorname{sign}(x_f) = \operatorname{sign}(\lambda)\operatorname{sign}(x)_f = \operatorname{sign}(\lambda)X_f.$$

Hence, e is parallel to f in $\operatorname{sign}(\mathcal{V})$. □

Example 4.34. We consider the following matrix:

```
sage: C = matrix([[2,0,0,0,-1,-3,0],[0,0,1,-2,0,0,0],[0,0,2,-4,0,0,1]])
sage: C
[ 2  0  0  0 -1 -3  0]
[ 0  0  1 -2  0  0  0]
[ 0  0  2 -4  0  0  1]
```

Here, we are interested in the vector space generated by the rows of this matrix. Therefore, it suffices to compute the parallel classes of the rows.

```
sage: parallel_classes(C)
[[0, 4, 5], [1], [2, 3], [6]]
```

Now, we compute the covectors of the corresponding oriented matroid.

```
sage: ccC = cocircuits_from_matrix(C)
sage: ccC
[(00+-000), (-000++0), (000000+), (00-+000), (+000--0), (000000-)]
sage: cvC = covectors_from_cocircuits(ccC)
sage: cvC
[(0000000), (00+-000), (-000++0), (000000+), (00-+000), (+000--0),
   (000000-), (00+-00-), (-000++-), (00-+00-), (+000---), (+0+----),
   (+000--+), (+0-+---), (-0-+++-), (+0-+--+), (+0+---+), (-0-++++),
   (-0+-+++), (-0+-++-), (-000+++), (00-+00+), (00+-00+), (+0+---0),
   (+0-+--0), (-0-+++0), (-0+-++0)]
```

Hence, the oriented matroid has the following parallel classes:

```
sage: parallel_classes(cvC)
[[0, 4, 5], [1], [2, 3], [6]]
```

These are the same parallel classes as we have obtained before.

Note that in order to compute the parallel classes of an oriented matroid, it suffices to compute the parallel classes of the cocircuits. Furthermore, the parallel classes of the tope set are the same as the parallel classes of the oriented matroid.

Lemma 4.35. *Let $e, f \in E$. The following are equivalent:*

(i) *The elements e and f are parallel in $\mathcal{F}$.*

(ii) *The elements e and f are parallel in $\mathcal{D}$.*

(iii) *The elements e and f are parallel in $\mathcal{T}$.*

Proof. The direction from (i) to (ii) is clear.

Assume that (ii) holds. Hence, there is $\Lambda \in \{-, +\}$ for every $V \in \mathcal{D}$ such that $V_e = \Lambda V_f$. Now, let $V^1, V^2 \in \mathcal{D}$. If $V^1_e = 0$, then by Lemma 4.27, $V^1_f = 0$ as well. Therefore,

$$(V^1 \circ V^2)_e = V^2_e = \Lambda V^2_f = \Lambda (V^1 \circ V^2)_f.$$

On the other hand, if $V^1_e \neq 0$, then by the same argument $V^1_f \neq 0$. Hence,

$$(V^1 \circ V^2)_e = V^1_e = \Lambda V^1_f = \Lambda (V^1 \circ V^2)_f.$$

Thus, e is parallel to f for every composition of two cocircuits. Consequently, e and f are parallel for every (finite) composition of cocircuits. With Corollary 4.15, we obtain (iii).

Finally, assume that (iii) holds and let $e, f \in E$ such that e and f are parallel in $\mathcal{T}$. Let $X \in \mathcal{F}$. By Proposition 4.22, we have $X \circ T \in \mathcal{F}$ for all $T \in \mathcal{T}$. Assume that $X_e = 0$ and $X_f \neq 0$. Since $X_f \neq 0$, every tope satisfies $T_f \neq 0$. Furthermore, Lemma 4.27 implies that every tope satisfies $T_e \neq 0$. Now, let $T \in \mathcal{T}$. We define $Y := X \circ T \in \mathcal{T}$ and $Z := X \circ (-T) \in \mathcal{T}$. Observe that

$$Y_e = (X \circ T)_e = -(X \circ (-T))_e = -Z_e \neq 0$$

and

$$Y_f = (X \circ T)_f = (X \circ (-T))_f = Z_f \neq 0.$$

Consequently, if $Y_e = Y_f$, then $Z_e = -Z_f$ and if $Y_e = -Y_f$, then $Z_e = Z_f$. Therefore, e and f are not parallel in $\mathcal{T}$ which is a contradiction.
If $X_e = X_f = 0$, then e and f are parallel for X. Therefore, assume that $X_e, X_f \neq 0$. Now, since e is parallel to f in $\mathcal{T}$, let $\Lambda \in \{-,+\}$ such that for all $T \in \mathcal{T}$, we have $T_e = \Lambda T_f$ and let $T \in \mathcal{T}$. By Proposition 4.22, we have $X \circ T \in \mathcal{T}$ and therefore $(X \circ T)_e = \Lambda(X \circ T)_f$. Now, if $X_e = -\Lambda X_f \neq 0$, then

$$(X \circ T)_e = X_e = -\Lambda X_f = -\Lambda(X \circ T)_f \neq 0$$

which is a contradiction. Hence, $X_e = \Lambda X_f$. Consequently, we obtain that e is parallel to f for X and therefore (i) holds. □

Proposition 4.36. *The parallel classes of $\mathcal{F}$ are the same as the parallel classes of the tope set $\mathcal{T}$ and the set of cocircuits $\mathcal{D}$.*

Proof. The result follows immediately from Lemma 4.35. □

Example 4.37. We continue with Example 4.34. Thus, we compute the parallel classes of the cocircuits.

```
sage: parallel_classes(ccC)
[[0, 4, 5], [1], [2, 3], [6]]
```

We have already obtained these parallel classes in Example 4.34.
Now, we compute the topes.

```
sage: tC = topes_from_cocircuits(ccC)
sage: tC
[(+0+----), (+0-+---), (+0+---+), (+0-+--+), (-0+-+++), (-0-++++),
   (-0-+++-), (-0+-++-)]
```

The topes have the following parallel classes:

```
sage: parallel_classes(tC)
[[0, 4, 5], [1], [2, 3], [6]]
```

These are the same parallel classes as we have computed previously.

4.3.2 Rank and dimension

An important property of the big face lattice is the so called Jordan-Dedekind chain property. The following proposition can be found in [Fuk04] (Proposition 1.2).

Proposition 4.38. *The big face lattice $\tilde{\mathcal{F}}$ satisfies the* Jordan-Dedekind chain property*, that is, every maximal chain between two comparable faces of $\tilde{\mathcal{F}}$ has the same length.*

With Proposition 4.38, we obtain the notion of a rank for the covectors of an oriented matroid. That way, the zero sign vector will receive rank 0 and by going up one step in the big face lattice, we increase the rank by 1. Because of the Jordan-Dedekind chain property, the rank of each covector will be uniquely defined.

Definition 4.39. Let $X \in \mathcal{F}$. The *rank* $\operatorname{rank}(X)$ of a face $X \in \mathcal{F}$ is the length of any maximal chain of $\mathcal{F}$ from $\mathbf{0}$ to X. The *dimension* $\dim(\mathcal{M})$ of an oriented matroid is the rank of any tope of $\mathcal{M}$. The *dimension* of a face $X \in \mathcal{F}$ is $\dim(X) = \operatorname{rank}(X) - 1$.
We denote by $\mathcal{F}_i \subseteq \mathcal{F}$ the set of faces with dimension i in $\mathcal{F}$ for $i \in \{-1, 0, \ldots, \dim(\mathcal{M})\}$.

From Definition 4.39, we obtain that the rank of the cocircuits is 1 and the dimension is 0. The zero covector has rank 0 and dimension -1.

4.3.3 Deletion

By deleting specific components of every covector, we obtain a new oriented matroid, that is, the deletion minor.

Definition 4.40. Let $\mathcal{S} \subseteq \{-, 0, +\}^E$ and let $R \subseteq E$. The *deletion* of $\mathcal{S}$ is defined by

$$\mathcal{S} \setminus R := \{X \setminus R \mid X \in \mathcal{S}\}.$$

We define the *deletion minor of an oriented matroid* $\mathcal{M}$ *w.r.t.* R to be the pair

$$\mathcal{M} \setminus R := (E \setminus R, \mathcal{F} \setminus R).$$

Proposition 4.41. *Deletion minors of oriented matroids are oriented matroids.*

Proof. Since $\mathbf{0} \in \mathcal{F}$, we have $\mathbf{0} \setminus R \in \mathcal{F} \setminus R$. Hence, (F0) is satisfied.

Let $X \in \mathcal{F}$. Then, $X \setminus R \in \mathcal{F} \setminus R$. Since $-X \in \mathcal{F}$, we obtain

$$-(X \setminus R) = -X \setminus R \in \mathcal{F} \setminus R$$

and therefore (F1) is fulfilled.

Now, let $X, Y \in \mathcal{F}$. Hence, $X \setminus R, Y \setminus R \in \mathcal{F} \setminus R$. Since $X \circ Y \in \mathcal{F}$, we have

$$(X \setminus R) \circ (Y \setminus R) = (X \circ Y) \setminus R \in \mathcal{F} \setminus R.$$

Thus, (F2) holds.

Finally, let $X, Y \in \mathcal{F}$ and $e \in D(X, Y) \setminus R$. Therefore, $X \setminus R, Y \setminus R \in \mathcal{F} \setminus R$ and $e \in D(X \setminus R, Y \setminus R)$. By (F3) of $\mathcal{M}$, there is $Z \in \mathcal{F}$ with $Z_e = 0$ and for all $f \in D \setminus D(X, Y)$, we have $Z_f = (X \circ Y)_f$. Hence, $Z \setminus R \in \mathcal{F} \setminus R$ and satisfies $(Z \setminus R)_e = 0$. Furthermore, for all

$$f \in (E \setminus D(X, Y)) \setminus R = (E \setminus R) \setminus D(X \setminus R, Y \setminus R),$$

we have

$$(Z \setminus R)_f = ((X \circ Y) \setminus R)_f = ((X \setminus R) \circ (Y \setminus R))_f.$$

Thus, (F3) is satisfied for $\mathcal{M} \setminus R$ which is therefore an oriented matroid. □

Deletion is available in the package [`sign_vectors`].

```
sage: from sign_vectors import deletion
```

Example 4.42. We consider the matrix

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

from Example 4.12. In Example 4.17, we have obtained the following covectors:

```
sage: cvB
```

```
[(0000), (+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-),
   (+0+-), (0---), (0-+-), (-0+-), (0++-), (0+++), (++++), (0--+),
   (0+-+), (-+++), (---+), (--+-), (-+-+), (-++-), (+++-), (----),
   (+---), (+--+), (++-+), (+-+-), (-0-+), (+0-+), (++0+), (-+0+),
   (--0-), (+-0-), (+++0), (-++0), (---0), (+--0)]
```

The covectors of the corresponding deletion minor with respect to the set $R := \{2\}$ are:

```
sage: deletion(cvB, [2])
[(000), (+00), (0-0), (0--), (00+), (-00), (0+0), (0++), (00-), (+0-),
   (-0-), (0+-), (+++), (0-+), (-++), (--+), (---), (-+-), (++-), (+--),
   (+-+), (-0+), (+0+), (++0), (-+0), (--0), (+-0)]
```

Deletion can also be defined for real subspaces. Note that the function `deletion` can also be applied to a matrix. In this case, the result will be a list of vectors. However, the same results can also be directly obtained without the package.

Example 4.43. We consider the vector space generated by the rows of the matrix

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}.$$

Now, we delete the third column, that is, the column with index 2. That way, we obtain the vector space generated by the vectors:

```
sage: deletion(B, [2])
[(1, 0, 0), (0, 1, 2), (0, 0, -1)]
```

4.3.4 Contraction

The next concept lets us create a new oriented matroid by taking only the covectors that have zeros at specific components.

Definition 4.44. Let $\mathcal{S} \subseteq \{-, 0, +\}^E$ and let $R \subseteq E$. The *contraction* of $\mathcal{S}$ is defined by

$$\mathcal{S}/R := \{X \setminus R \mid X \in \mathcal{S} \text{ and } R \subseteq X^0\}.$$

We define the *contraction minor of an oriented matroid* $\mathcal{M}$ *w.r.t.* R to be the pair

$$\mathcal{M}/R := (E \setminus R, \mathcal{F}/R).$$

Similar to deletion minors, we show that contraction minors are oriented matroids.

Proposition 4.45. *Contraction minors of oriented matroids are oriented matroids.*

Proof. Let $R \subseteq E$. Since $R \subseteq E = \mathbf{0}^0$, we have $\mathbf{0} \setminus R \in \mathcal{M}$. Hence, (F0) is fulfilled for $\mathcal{M}/R$.

Next, let $X \in \mathcal{F}$ with $R \subseteq X^0$. Then, also $R \subseteq (-X)^0$. Consequently, (F1) is fulfilled for $\mathcal{M}/R$.

Now, let $X, Y \in \mathcal{F}$ with $R \subseteq X^0, Y^0$. Then, also $R \subseteq (X \circ Y)^0$. Thus, (F2) is fulfilled for $\mathcal{M}/R$.

Finally, let $X, Y \in \mathcal{F}$ and $e \in D(X, Y)$ such that $R \subseteq X^0, Y^0$. By (F3), there is $Z \in \mathcal{F}$ with $Z_e = 0$ and for all $f \in E \setminus D(X, Y)$, we have $Z_f = (X \circ Y)_f$. Since $R \subseteq E \setminus D(X, Y)$, we have $Z_f = (X \circ Y)_f = 0$ for $f \in R$ and consequently $R \subseteq Z^0$. Hence, (F3) is fulfilled for $\mathcal{M}/R$ and therefore, $\mathcal{M}/R$ is an oriented matroid. □

The contraction can be computed with the package [sign_vectors].

```
sage: from sign_vectors import contraction
```

Example 4.46. We take the matrix

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

from Example 4.12. In Example 4.17, we obtained the following covectors:

```
sage: cvB
[(0000), (+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-),
   (+0+-), (0---), (0-+-), (-0+-), (0++-), (0+++), (++++), (0--+),
   (0+-+), (-+++), (---+), (--+-), (-+-+), (-++-), (+++-), (----),
   (+---), (+--+), (++-+), (+-+-), (-0-+), (+0-+), (++0+), (-+0+),
   (--0-), (+-0-), (+++0), (-++0), (---0), (+--0)]
```

Now, we compute the corresponding contraction minor with respect to $R := \{3\}$.

```
sage: contraction(cvB, [3])
[(000), (+00), (0--), (-00), (0++), (+++), (-++), (---), (+--)]
```

With the optional argument `keep_components=True`, we can keep the components of the covectors that are zero on R.

```
sage: contraction(cvB, [3], keep_components=True)
[(0000), (+000), (0--0), (-000), (0++0), (+++0), (-++0), (---0), (+--0)]
```

We obtain the same contraction minor $\mathcal{M}/R$ if we apply contraction only to the cocircuits of $\mathcal{M}$ and use them to construct each covector of the oriented matroid $\mathcal{M}/R$.

Proposition 4.47. *The set*

$$\mathcal{D}/R := \{X \setminus R \mid X \in \mathcal{D} \text{ and } R \subseteq X^0\}$$

is the set of cocircuits of the contraction minor $\mathcal{M}/R$.

Proof. The elements in $\mathcal{D}/R$ generate every covector in $\mathcal{F}/R$. □

Example 4.48. We continue Example 4.46. The cocircuits of the oriented matroid corresponding to the matrix B are:

```
sage: ccB
[(+000), (0--0), (0-0-), (00-+), (-000), (0++0), (0+0+), (00+-)]
```

Next, we apply contraction to those cocircuits.

```
sage: con_ccB = contraction(ccB, [3])
sage: con_ccB
[(+00), (0--), (-00), (0++)]
```

Now, we use these cocircuits to compute the contraction minor.

```
sage: covectors_from_cocircuits(con_ccB)
```

```
[(000), (+00), (0--), (-00), (0++), (+++), (-++), (---), (+--)]
```

These are the same covectors as we determined in Example 4.46.

Contraction can also be applied to vector spaces.

Example 4.49. We consider the vector space generated by the rows of the matrix

$$B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

from Example 4.12. Now, we compute the elementary vectors corresponding to the rows of this matrix and apply `contraction` with respect to the set $R = \{3\}$ to them.

```
sage: evB = elementary_vectors(B)
sage: evB
[(2, 0, 0, 0), (0, -1, -2, 0), (0, -1, 0, -2), (0, 0, -1, 1)]
sage: contraction(evB, [3])
[(2, 0, 0), (0, -1, -2)]
```

The resulting elementary vectors correspond to the cocircuits of the contraction minor from Example 4.48. Hence, the oriented matroid corresponding to the contraction of a vector space is the same as the contraction minor of the initial vector space.

In contrast to deletion, applying `contraction` directly to the matrix B leads to a different vector space.

```
sage: contraction(B, [3])
[(1, 0, 0)]
```

4.3.5 Lower Faces

The next lemma is based on Lemma 0.7.6 in [Fin01] and connects the conformal relation to parallel classes of a contraction minor.

Lemma 4.50. *Let $X, Y \in \mathcal{F}$ with $X \prec Y$. The following statements are equivalent:*

(i) *$X^0 \setminus Y^0$ is a parallel class of $\mathcal{F}/Y^0$.*

(ii) *X is covered by Y, that is, there is no $Z \in \mathcal{F}$ with $X \prec Z \prec Y$.*

Proof. Assume that $S := X^0 \setminus Y^0$ is not a parallel class of $\mathcal{F}/Y^0$. By Lemma 4.27, there are $e, f \in S$ and $Z \in \mathcal{F}$ such that $Y^0 \subseteq Z^0$ and exactly one of Z_e and Z_f is zero. Without loss of generality, assume that $Z_e = 0$ and $Z_f \neq 0$. We can further assume that $Z_f = Y_f$. Otherwise, take $-Z$ instead of Z.
With Proposition 3.19 and $X_f = 0 \neq Z_f$, we obtain $X \prec X \circ Z$. Furthermore, $Y^0 \subseteq X^0$ and $Y^0 \subseteq Z^0$ implies that $Y^0 \subseteq (X \circ Z)^0$. Now, we have

$$0 = X_e = Z_e = (X \circ Z)_e \neq Y_e.$$

If in addition, $D := D(Y, X \circ Z) = \emptyset$, we obtain $X \circ Z \prec Y$ and thus

$$X \prec X \circ Z \prec Y.$$

Conversely, if $D \neq \emptyset$, we apply conformal elimination (F3c) to $Y, X \circ Z$ and D. Then, there are $g \in D$ and $U \in \mathcal{F}$ with $U_g = 0$, $U_D \preceq Y_D$ and $U \setminus D = (Y \circ X \circ Z) \setminus D$. Since $Y^0 \subseteq (X \circ Z)^0$, we have

$$U \setminus D = (Y \circ X \circ Z) \setminus D = Y \setminus D.$$

Since $X_f = 0$ and $Z_f = Y_f = (X \circ Y)_f$, we have $f \notin D$. Hence,

$$0 = X_f \neq (X \circ U)_f = U_f = Y_f.$$

Consequently, $X \prec X \circ U$. Now, $g \in D \subseteq \underline{Y} \setminus \underline{X}$ and therefore

$$0 = X_g = U_g = (X \circ U)_g \neq Y_g$$

and we obtain $X \circ U \prec Y$. In summary, we have

$$X \prec X \circ U \prec Y.$$

For the remaining direction, assume that there is $Z \in \mathcal{F}$ with $X \prec Z \prec Y$. Then, there are $e, f \in S := X^0 \setminus Y^0$ such that $e \in Z^0$ and $f \notin Z^0$. Therefore, by Lemma 4.27, S is not a parallel class of $\mathcal{F}/Y^0$. □

For the next results, we use the following notation. Let $X \in \mathcal{F}$ and define

$$\mathcal{F}(\underline{X}) := \{Z \setminus X^0 \mid Z \in \mathcal{F} \text{ and } \underline{Z} = \underline{X}\}.$$

This set can be constructed by first restricting each covector of $\mathcal{F}$ to $\underline{X}$ and then taking only those sign vectors that have no zero component.
We will show that this set is the tope set of a special contraction minor.

Lemma 4.51. *For a covector $X \in \mathcal{F}$, the set $\mathcal{F}(\underline{X})$ is the tope set of the contraction minor $\mathcal{F}/X^0$.*

Proof. Let $Y \in \mathcal{F}$. It follows immediately that $Y^0 = X^0$ is equivalent to $\underline{Y} = \underline{X}$ which is further equivalent to $Y \setminus X^0 \in \mathcal{F}(\underline{X})$. Furthermore, we have $Y^0 = X^0$ if and only if $Y \setminus X^0$ is a tope of $\mathcal{F}/X^0$. Hence, $\mathcal{F}(\underline{X})$ is the tope set of $\mathcal{F}/X^0$. □

With Lemma 4.51, we obtain the following lemma which is essentially Lemma 1.5.5 from [Fin01].

Lemma 4.52. *Let $X \in \mathcal{F}$. Then, the parallel classes of $\mathcal{F}/X^0$ and $\mathcal{F}(\underline{X})$ are the same.*

Proof. The result follows immediately from Proposition 4.36 and Lemma 4.51. □

The next lemma is Lemma 1.5.6 from [Fin01]. This lemma lets us characterise the faces of an oriented matroid.

Lemma 4.53. *Let $d := \dim(\mathcal{M})$. For any $i \in \{0, \dots, d-1\}$, we have $Z \in \mathcal{F}_i$ if and only if there are $X, Y \in \mathcal{F}_{i+1}$ with $\underline{X} = \underline{Y}$ and $D := D(X, Y)$ is a parallel class of $\mathcal{F}(\underline{X})$ and $Z \setminus D = X \setminus D$ and $Z_D = \mathbf{0}$.*

Proof. Assume that there are $X, Y \in \mathcal{F}_{i+1}$ with $\underline{X} = \underline{Y}$ and assume that $D := D(X, Y) \neq \emptyset$ is a parallel class of $\mathcal{F}(\underline{X})$. Next, apply conformal elimination (F3c) to X, Y and D. Therefore, there is $e \in D$ and $Z \in \mathcal{F}$ with

$$Z_e = 0, \quad Z_D \preceq X_D \quad \text{and} \quad Z \setminus D = (X \circ Y) \setminus D.$$

Since $\underline{X} = \underline{Y}$, we obtain $X \circ Y = X$. Consequently,

$$Z \setminus D = (X \circ Y) \setminus D = X \setminus D.$$

Hence, $Z \prec X$ and therefore $X^0 \subseteq Z^0$. By Lemma 4.52, D is a parallel class of $\mathcal{F}/X^0$ and since $Z_e = 0$, we obtain with Lemma 4.27 that $Z_D = \mathbf{0}$.
Now, we have $Z^0 = D \cup X^0$ and consequently $D = Z^0 \setminus X^0$. With Lemma 4.50, X covers Z. Thus, $Z \in \mathcal{F}_i$.

To show the other direction, let $Z \in \mathcal{F}_i$. Since $i < d$, there is $X \in \mathcal{F}_{i+1}$ such that $Z \prec X$. Next, define $Y := Z \circ (-X)$. Hence, $\underline{X} = \underline{Y}$ and $Y \in \mathcal{F}_{i+1}$. Furthermore, let $D := D(X, Y)$. Then, $Z \setminus D = X \setminus D$ and $Z_D = \mathbf{0}$. Similar to the first direction, we have $D = Z^0 \setminus X^0$. By Lemma 4.50, D is a parallel class of $\mathcal{F}/X^0$ and with Lemma 4.52, D is a parallel class of $\mathcal{F}(\underline{X})$. □

Lemma 4.53 gives rise to the algorithm `lower_faces`. The idea is to take two elements of $\mathcal{F}_{i+1}$ with the same support. Then, we use those two elements to construct an element of $\mathcal{F}_i$. Therefore, we can apply this to the whole set $\mathcal{F}_{i+1}$ and obtain $\mathcal{F}_i$ as a consequence. Continuing this procedure leads to the algorithm `face_enumeration` which is proposed in [FST91]. Here, we start with the tope set and apply `lower_faces` several times to obtain all covectors separated by their rank.
For further details on the algorithms `lower_faces` and `face_enumeration`, consult [Fin01] and [FST91].
Implementations of both algorithms are available in the package [`sign_vectors`].

4.3.6 Examples

We conclude this section with several examples.

Example 4.54. Again, we consider the matrix

$$A := \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \end{pmatrix}$$

from Example 4.11, where we have received the following topes:

```
sage: tA
[(---), (-+-), (++-), (+++), (--+), (+-+)]
```

The function `face_enumeration` from the package computes all faces if we apply it to the tope set.

```
sage: feA = face_enumeration(tA)
sage: feA
[[(000)], [(-0-), (--0), (0+-), (++0), (+0+), (0-+)], [(---), (-+-),
   (++-), (+++), (--+), (+-+)]]
```

The first entry of the resulting list is just the zero sign vector $\mathbf{0}$ which is the only element with rank 0.

```
sage: feA[0]
[(000)]
```

The zero sign vector is covered by the cocircuits. Hence, the next entry consists of the elements of rank 1.

```
sage: feA[1]
[(-0-), (--0), (0+-), (++0), (+0+), (0-+)]
```

Finally, we arrive at the last entry of the list. In our case, these are the topes. Here, those elements have rank 2.

```
sage: feA[2]
[(---), (-+-), (++-), (+++), (--+), (+-+)]
```

We conclude that the oriented matroid corresponding to these covectors has dimension 2.

The package also offers the function `covectors_from_topes`. Here, `face_enumeration` is applied to a list of topes to compute a list of every covector of the corresponding oriented matroid.

```
sage: covectors_from_topes(tA)
[(000), (-0-), (--0), (0+-), (++0), (+0+), (0-+), (---), (-+-), (++-),
    (+++), (--+), (+-+)]
```

Example 4.55. Now, we consider the matrix

$$B := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & -1 \end{pmatrix}$$

which we have defined in Example 4.12. Previously, we have obtained the topes:

```
sage: tB
[(++++), (-+++), (++-+), (-+-+), (+--+), (---+), (+++-), (-++-), (----),
    (--+-), (+---), (+-+-)]
```

We apply `face_enumeration` to this list of topes.

```
sage: feB = face_enumeration(tB)
sage: feB
[[(0000)], [(0+0+), (0++0), (00-+), (0--0), (00+-), (0-0-), (+000),
    (-000)], [(0+++), (++0+), (+++0), (-+0+), (-++0), (0+-+), (+0-+),
    (-0-+), (0--+), (+--0), (---0), (0++-), (+0+-), (-0+-), (0---),
    (--0-), (0-+-), (+-0-)], [(++++), (-+++), (++-+), (-+-+), (+--+),
    (---+), (+++-), (-++-), (----), (--+-), (+---), (+-+-)]]
```

The resulting list consists of four elements. Hence, the covectors corresponding to the rows of B have rank 0, 1, 2 and 3.
Again, the zero vector is the only element of rank 0.

```
sage: feB[0]
[(0000)]
```

Next, we arrive at the cocircuits which have rank 1.

```
sage: feB[1]
[(0+0+), (0++0), (00-+), (0--0), (00+-), (0-0-), (+000), (-000)]
```

The next list consists of covectors of rank 2.

```
sage: feB[2]
[(0+++), (++0+), (+++0), (-+0+), (-++0), (0+-+), (+0-+), (-0-+), (0--+),
    (+--0), (---0), (0++-), (+0+-), (-0+-), (0---), (--0-), (0-+-), (+-0-)]
```

The final list is the list of topes. Here, the topes have rank 3.

```
sage: feB[3]
[(++++), (-+++), (++-+), (-+-+), (+--+), (---+), (+++-), (-++-), (----),
   (--+-), (+---), (+-+-)]
```

Therefore, the resulting oriented matroid has dimension 3.

From the previous two examples, one might think that there is a direct correspondence between the number of zeros and the rank of a covector. That does not hold in general. Loops and parallel elements might lead to additional zero entries. In the next example, we will consider such an oriented matroid.

Example 4.56. Now, we consider the matrix

$$C = \begin{pmatrix} 2 & 0 & 0 & 0 & -1 & -3 & 0 \\ 0 & 0 & 1 & -2 & 0 & 0 & 0 \\ 0 & 0 & 2 & -4 & 0 & 0 & 1 \end{pmatrix}$$

from Example 4.34. One can already see from the matrix C that every covector has a zero entry at the second component. Therefore, this component is a loop.

We have already computed the list of topes in Example 4.37.

```
sage: tC
[(+0+----), (+0-+---), (+0+---+), (+0-+--+), (-0+-+++), (-0-++++),
   (-0-+++-), (-0+-++-)]
```

Next, we apply `face_enumeration` to the list of topes.

```
sage: tC = topes_from_cocircuits(ccC)
sage: feC = face_enumeration(tC)
sage: feC
[[(0000000)], [(000000-), (00+-000), (00-+000), (000000+), (+000--0),
   (-000++0)], [(00+-00-), (+000---), (+0+---0), (00-+00-), (+0-+--0),
   (00+-00+), (+000--+), (00-+00+), (-000+++), (-0+-++0), (-0-+++0),
   (-000++-)], [(+0+----), (+0-+---), (+0+---+), (+0-+--+), (-0+-+++),
   (-0-++++), (-0-+++-), (-0+-++-)]]
```

As one can see from the output, we obtain four lists of covectors. Hence, there are covectors of four different ranks and the topes have rank 3.

The first list (with index 0) consists only of the zero vector $\mathbf{0}$.

```
sage: feC[1]
[(000000-), (00+-000), (00-+000), (000000+), (+000--0), (-000++0)]
```

Observe that there are cocircuits with four, five and even six zero entries.

```
sage: feC[2]
[(00+-00-), (+000---), (+0+---0), (00-+00-), (+0-+--0), (00+-00+),
   (+000--+), (00-+00+), (-000+++), (-0+-++0), (-0-+++0), (-000++-)]
```

Consequently, there are covectors of rank 2 with two, three and four zero components.

```
sage: feC[3]
```

```
[(+0+----), (+0-+---), (+0+---+), (+0-+--+), (-0+-+++), (-0-++++),
   (-0-+++-), (-0+-++-)]
```

The topes have all one zero entry, that is, the second component as we have mentioned above.

As we have seen in the previous examples, to apply `face_enumeration`, we need to compute the topes first. Furthermore, in order to obtain the topes, we need to compute the cocircuits first. The package [`sign_vectors`] offers shortcuts to compute topes and covectors from the cocircuits.

Example 4.57. We consider the matrix

$$A = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & -1 \end{pmatrix}$$

from Example 4.11. We can directly compute the covectors and topes from A by applying the appropriate functions.

```
sage: covectors_from_matrix(A)
[(000), (--0), (-0-), (0-+), (++0), (+0+), (0+-), (---), (-+-), (++-),
   (+++), (--+), (+-+)]
sage: topes_from_matrix(A)
[(---), (-+-), (++-), (+++), (--+), (+-+)]
```

By default, `covectors_from_matrix` applies first `cocircuits_from_matrix` to A. Afterwards, the function `covectors_from_cocircuits` is applied to the computed list of cocircuits. We can pass `algorithm='fe'` if we want to apply `topes_from_cocircuits` and then `face_enumeration` instead of `covectors_from_cocircuits`.

```
sage: covectors_from_matrix(A, algorithm='fe')
[(000), (-0-), (--0), (0+-), (++0), (+0+), (0-+), (---), (-+-), (++-),
   (+++), (--+), (+-+)]
```

By specifying `separate=True`, the resulting covectors can be split into lists of covectors with the same rank.

```
sage: covectors_from_matrix(A, algorithm='fe', separate=True)
[[(000)], [(-0-), (--0), (0+-), (++0), (+0+), (0-+)], [(---), (-+-),
   (++-), (+++), (--+), (+-+)]]
```

We can also compute the cocircuits from the topes by applying `face_enumeration` and taking the second list from the output. This is the way how the function `cocircuits_from_topes` is set up.

```
sage: tA = topes_from_matrix(A)
sage: cocircuits_from_topes(tA)
[(-0-), (--0), (0+-), (++0), (+0+), (0-+)]
```

4.4 Duality

The set $\mathcal{F}^\perp$ is called the *orthogonal space* (or *dual space*) of $\mathcal{F}$. There are many important results about duality of oriented matroids. However, we will not use these facts in an essential way. In the following, we summarize some of those results. For the proofs, we refer the reader to [Fin01].

The first result gives rise to the dual oriented matroid.

Theorem 4.58. *The pair $(E, \mathcal{F}^\perp)$ is an oriented matroid.*

Definition 4.59. We call $\mathcal{M}^\perp := (E, \mathcal{F}^\perp)$ the dual of $\mathcal{M}$.

Similar to real subspaces, the dual of the dual oriented matroid is the original oriented matroid.

Proposition 4.60. *For every oriented matroid $\mathcal{M}$, we have $\mathcal{M}^{\perp\perp} = \mathcal{M}$.*

If we have given the rank of an oriented matroid, we can determine the rank of the dual analogously as for the dimension of real subspaces.

Corollary 4.61. *The rank of the dual is*

$$\operatorname{rank}(\mathcal{M}^\perp) = |E| - \operatorname{rank}(\mathcal{M}).$$

Example 4.62. We consider the following matrix:

```
sage: D = matrix([[1,0,-2,1,0],[0,1,2,-1,1]])
sage: D
[ 1  0 -2  1  0]
[ 0  1  2 -1  1]
```

Now, we compute the covectors of the oriented matroid corresponding to this matrix.

```
sage: feD = covectors_from_matrix(A, algorithm='fe', separate=True)
sage: feD
[[(000)], [(-0-), (--0), (0+-), (++0), (+0+), (0-+)], [(---), (-+-),
   (++-), (+++), (--+), (+-+)]]
```

Hence, the dimension of the oriented matroid is 2.

The matrix corresponding to the dual oriented matroid is the kernel matrix

$$\begin{pmatrix} 1 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 2 & 0 \end{pmatrix}.$$

If we pass the optional argument `kernel=True` to the function `cocircuits_from_matrix`, we obtain the cocircuits of the dual oriented matroid.

```
sage: cocircuits_from_matrix(D, kernel=True)
[(+-+00), (-+0+0), (0-00+), (00++0), (-0-0+), (+00--), (-+-00), (+-0-0),
   (0+00-), (00--0), (+0+0-), (-00++)]
```

The functions `covectors_from_matrix` and `topes_from_matrix` also support the optional argument `kernel=True`.

```
sage: covectors_from_matrix(D, kernel=True)
[(00000), (+-+00), (-+0+0), (0-00+), (00++0), (-0-0+), (+00--), (-+-00),
   (+-0-0), (0+00-), (00--0), (+0+0-), (-00++), (+-+++), (-+0++),
   (--0++), (-0+++), (-0-++), (-+-++), (+-0-+), (-+0+-), (-0--+),
   (+0++-), (+-++-), (-+++-), (+0+--), (-+-+-), (+-+--), (++++-),
   (+0---), (+---+), (-+---), (+----), (++---), (-+--+), (+++--),
   (+-+-+), (-++++), (--+++), (---++), (----+), (++0--), (+-0--),
   (+-+0-), (+-+0+), (-+-0-), (+++0-), (-+-0+), (---0+), (+-+-0),
   (-+-+0), (0---+), (-+--0), (+---0), (0+---), (0+++-), (0-+++),
   (+-++0), (-+++0)]
sage: topes_from_matrix(D, kernel=True)
[(+-+++), (-+-++), (+-++-), (-+++-), (-+-+-), (+-+--), (++++-), (+---+),
   (-+---), (+----), (++---), (+-+-+), (+++--), (----+), (-+--+),
   (---++), (-++++), (--+++)]
```

Now, we compute the covectors of the dual oriented matroid.

```
sage: feDd = covectors_from_matrix(A, kernel=True, algorithm='fe',
    separate=True)
sage: feDd
[[(000)], [(-++), (+--)]]
```

Therefore, the dimension of the dual oriented matroid is 1.

5 Applications to bijectivity of exponential maps

In [MHR19], there are several conditions for simultaneous injectivity and bijectivity of exponential maps that can be checked algorithmically. The corresponding algorithms are implemented in the package [bijectivity_exponential_maps]. In this chapter, we will use the variable names and several notations from [MHR19].

5.1 General definitions

From now on, $\mathbb{R}_{>0}$ stands for the positive real numbers. For a vector $x \in \mathbb{R}^n$, we write $x > 0$ to specify that $x \in \mathbb{R}_{>0}$. Similarly, we use $x \leq 0$ to denote a real vector without positive components. For two real vectors $x, y \in \mathbb{R}^n$, we write $x \cdot y$ to denote their scalar product.

Definition 5.1. A *cone* C is a subset of $\mathbb{R}^n$ such that for all $x, y \in C$ and $\lambda \geq 0$, we have $x + y \in C$ and $\lambda x \in C$. By C°, we denote the interior of the cone C.

Let $W \in \mathbb{R}^{d \times n}$, $\tilde{W} \in \mathbb{R}^{d \times n}$ be matrices with $d \leq n$ and full rank. Additionally, let

$$C := \operatorname{cone} W \subseteq \mathbb{R}^d$$

be the cone generated by the columns of W. The cone C has non-empty interior C° since W has full rank. Furthermore, let $c \in \mathbb{R}^n_{>0}$. We define the polynomial map

$$\begin{aligned} f_c : \mathbb{R}^d_{>0} &\to C^\circ \subseteq \mathbb{R}^d, \\ x &\mapsto \left(\sum_{j=1}^{n} w_{ij} c_j \prod_{k=1}^{d} x_k^{\tilde{w}_{kj}} \right)_{i=1}^{d} \end{aligned}$$

and the exponential map

$$\begin{aligned} F_c : \mathbb{R}^d &\to C^\circ \subseteq \mathbb{R}^d, \\ x &\mapsto \sum_{i=1}^{n} c_i e^{\tilde{w}^i \cdot x} w^i. \end{aligned}$$

The question is whether the system

$$\sum_{j=1}^{n} w_{ij} c_j \prod_{k=1}^{d} x_k^{\tilde{w}_{kj}} = y_i, \quad \text{for } i = 1, \ldots, d$$

has a positive solution $x \in \mathbb{R}^d_{>0}$ for all $y \in C^\circ \subseteq \mathbb{R}^d$. This is equivalent to asking whether the polynomial map f_c is bijective. Furthermore, the exponential map F_c is bijective if and only if f_c is bijective. Consequently, it suffices to investigate bijectivity of the exponential map F_c.

The functions f_c and F_c are implemented in the package [bijectivity_exponential_maps].

```
sage: from bijectivity_exponential_maps import *
```

Example 5.2. We define some matrices and a vector c.

```
sage: var('x1, x2')
(x1, x2)
sage: x = [x1, x2]
sage: W = matrix([[1,0,-1],[0,1,-1]])
```

```
sage: W
[ 1  0 -1]
[ 0  1 -1]
sage: Wt = matrix([[1,0,-1],[0,1,0]])
sage: Wt
[ 1  0 -1]
[ 0  1  0]
sage: c = vector([1,2,4])
```

The command `f_pol` returns a function that takes a list or vector as argument. That way, we can apply f_c to points.

```
sage: fc = f_pol(W, Wt, c)
sage: fc(x)
(x1 - 4/x1, 2*x2 - 4/x1)
sage: fc([1,2])
(-3, 0)
```

We can also omit the argument `c`. In this case, `f_pol` uses the vector that is one on every component.

```
sage: fc = f_pol(W, Wt)
sage: fc(x)
(x1 - 1/x1, x2 - 1/x1)
sage: fc([1,2])
(0, 1)
```

Similarly, we can compute F_c.

```
sage: Fc = f_exp(W, Wt, c)
sage: Fc(x)
(-4*e^(-x1) + e^x1, -4*e^(-x1) + 2*e^x2)
sage: Fc([1,2])
(e - 4*e^(-1), 2*e^2 - 4*e^(-1))
```

Here, we can also omit `c` for `f_exp`.

```
sage: Fc = f_exp(W, Wt)
sage: Fc(x)
(-e^(-x1) + e^x1, -e^(-x1) + e^x2)
sage: Fc([1,2])
(e - e^(-1), e^2 - e^(-1))
```

For the remaining chapter, we define the subspaces

$$S := \ker(W) \subseteq \mathbb{R}^n \quad \text{and} \quad \tilde{S} := \ker(\tilde{W}) \subseteq \mathbb{R}^n.$$

5.2 Characterizing injectivity

In this section, we want to characterize injectivity of exponential maps. By Theorem 3.6 of [MR12], the map F_c is injective for all $c > 0$ if the intersection of the oriented matroids $\operatorname{sign}(S)$ and $\operatorname{sign}(\tilde{S}^\perp)$ contains only the zero sign vector. In addition, there is another equivalent condition that involves

the maximal minors of W and $\tilde{W}$, see [CGPS10]. Both results are summarized in Corollary 4 from [MHR19].

Theorem 5.3. *Let $S, \tilde{S}$ be subspaces of $\mathbb{R}^n$ of dimension $n-d$ with $d \leq n$. For every $W, \tilde{W} \in \mathbb{R}^{d\times n}$ with full rank d such that $S = \ker(W)$ and $\tilde{S} = \ker(\tilde{W})$, the following statements are equivalent:*

(1) *F_c is injective for all $c > 0$.*

(2) $\operatorname{sign}(S) \cap \operatorname{sign}(\tilde{S}^{\perp}) = \{\mathbf{0}\}$.

(3) $\det(W_I)\det(\tilde{W}_I) \geq 0$ *for all subsets $I \subseteq \{1, \ldots, n\}$ of cardinality d (or "≤ 0" for all I) and* $\det(W_I)\det(\tilde{W}_I) \neq 0$ *for some I.*

Condition (2) and (3) can be checked algorithmically. To investigate condition (2), we can use the package [`sign_vectors`] to compute the corresponding oriented matroids. Then, we compute their intersection. In order to check condition (3), we need to compute the maximal minors first and multiply corresponding minors. Here is another way of stating condition (3):

- Either

$$\det(W_I)\det(\tilde{W}_I) \geq 0$$

for all subsets $I \subseteq \{1, \ldots, n\}$ of cardinality d; or

$$\det(W_I)\det(\tilde{W}_I) \leq 0$$

for all subsets $I \subseteq \{1, \ldots, n\}$ of cardinality d.

5.2.1 Examples

Now, we consider some examples.

Example 5.4. Let $W, \tilde{W}$ be the following matrices.

```
sage: W = matrix([[1,0,-1],[0,1,-1]])
sage: W
[ 1  0 -1]
[ 0  1 -1]
sage: Wt = matrix([[1,0,-1],[0,1,0]])
sage: Wt
[ 1  0 -1]
[ 0  1  0]
```

We compute the corresponding oriented matroids.

```
sage: cvW = covectors_from_matrix(W, algorithm='fe')
sage: cvW
[(000), (0-+), (+-0), (+0-), (-0+), (-+0), (0+-), (+-+), (+--), (-++),
   (--+), (-+-), (++-)]
sage: cvWt = covectors_from_matrix(Wt, kernel=True, algorithm='fe')
sage: cvWt
[(000), (+0+), (-0-)]
```

The intersection of these oriented matroids consists only of the zero vector.
We can compute the intersection directly by applying the built in method `intersection`.

```
sage: set(cvW).intersection(cvWt)
{(000)}
```

Therefore, statement (2) is fulfilled for W and $\tilde{W}$ and consequently F_c is injective.
The package [bijectivity_exponential_maps] offers a usually more efficient way to check condition (2). Instead of computing the intersection, the corresponding function immediately returns when there is a non-zero vector in the intersection.

```
sage: cond_inj_intersection(W, Wt)
True
```

Note that `cond_inj_intersection` also works for matrices with different numbers of rows. Now, we want to check condition (3). First, observe that the minors of W and $\tilde{W}$ are:

```
sage: m1 = W.minors(2)
sage: m1
[1, -1, 1]
sage: m2 = Wt.minors(2)
sage: m2
[1, 0, 1]
```

Then, we multiply those minors component-wise.

```
sage: [m1[i]*m2[i] for i in range(len(m1))]
[1, 0, 1]
```

Hence, statement (3) is also satisfied. We can also check condition (3) by applying the following function.

```
sage: cond_inj_minors(W, Wt)
True
```

Now, let us consider an example where the corresponding function F_c is not injective.

Example 5.5. We consider the following matrices.

```
sage: W = matrix([[1,0,-1],[0,1,-1]])
sage: W
[ 1  0 -1]
[ 0  1 -1]
sage: Wt = matrix([[1,0,-1],[0,1,1]])
sage: Wt
[ 1  0 -1]
[ 0  1  1]
```

Next, we compute the corresponding oriented matroids.

```
sage: covectors_from_matrix(W, algorithm='fe', separate=True)
[[(000)], [(0-+), (+-0), (+0-), (-0+), (-+0), (0+-)], [(+-+), (+--),
   (-++), (--+), (-+-), (++-)]]
sage: covectors_from_matrix(Wt, kernel=True, algorithm='fe',
   separate=True)
[[(000)], [(+-+), (-+-)]]
```

Now, we check condition (2) with the corresponding function from the package.

```
sage: cond_inj_intersection(W, Wt)
False
```

Consequently, statement (2) does not hold for W and $\tilde{W}$ and the resulting map F_c is not injective. Furthermore, we obtain the following minors.

```
sage: m1 = W.minors(2)
sage: m1
[1, -1, 1]
sage: m2 = Wt.minors(2)
sage: m2
[1, 1, 1]

sage: [m1[i]*m2[i] for i in range(len(m1))]
[1, -1, 1]
```

The component-wise multiplication of the minors shows that statement (3) is not fulfilled. In addition, we apply the appropriate function from the package.

```
sage: cond_inj_minors(W, Wt)
False
```

Finally, we investigate an exponential map with parameters.

Example 5.6. Now, consider the following matrices.

```
sage: var('a,b')
(a, b)
sage: W = matrix([[1,0,-1],[0,1,-1]])
sage: W
[ 1  0 -1]
[ 0  1 -1]
sage: Wt = matrix([[1,0,a],[0,1,b]])
sage: Wt
[1 0 a]
[0 1 b]
```

The matrix $\tilde{W}$ contains the variables $a, b \in \mathbb{R}$. Consequently, we cannot compute the corresponding oriented matroids and check whether statement (2) is fulfilled for W and $\tilde{W}$.

On the other hand, we can still compute the minors of W and $\tilde{W}$, that is:

```
sage: m1 = W.minors(2)
sage: m1
[1, -1, 1]
sage: m2 = Wt.minors(2)
sage: m2
[1, b, -a]

sage: [m1[i]*m2[i] for i in range(len(m1))]
[1, -b, -a]
```

Therefore, statement (3) is satisfied if and only if $a \leq 0$ and $b \leq 0$.
The function `cond_inj_minors` also works for matrices with symbolic entries. In this case, it returns a system of inequalities.

```
sage: cond_inj_minors(W, Wt)
[-b >= 0, -a >= 0]
```

Consequently, by Theorem 5.3, F_c is injective if and only if $a \leq 0$ and $b \leq 0$.

5.3 Characterizing bijectivity

In this section, we consider bijectivity of exponential maps. For the next results, we introduce a notation for non-negative subsets of sign vectors. Let $T \subseteq \{-, 0, +\}^E$ be a subset. We define

$$T_\oplus := T \cap \{0, +\}^E.$$

Vectors with equal positive components play an important role for characterizing bijectivity of exponential maps. For this purpose, we introduce the corresponding positive sign vectors. We define for $\lambda \in \mathbb{R}_{>0}$ and $z \in \mathbb{R}^n$ the sign vector $\pi(\lambda, z)$ by

$$\pi(\lambda, z)_i := \begin{cases} + & \text{if } z_i = \lambda, \\ 0 & \text{else.} \end{cases}$$

The next definition is essential for characterizing bijectivity.

Definition 5.7. Let $S, \tilde{S}$ be subspaces of $\mathbb{R}^n$. The pair $(S, \tilde{S})$ is called *non-degenerate* if,

(i) for every $z \in \tilde{S}^\perp$, we have $z \leq 0$ or there is i with $z_i > 0$ and $\pi(z_i, z) \notin \operatorname{sign}(S)_\oplus$; or

(ii) for every cocircuit $\tilde{\tau} \in \operatorname{sign}(\tilde{S}^\perp)$, there is a positive cocircuit $\tau \in \operatorname{sign}(S^\perp)$ such that $\tilde{\tau}^0 \subseteq \tau^0$.

Finally, we will characterize bijectivity of F_c for all $c > 0$. This is the main result of [MHR19]. Recall that a *diffeomorphism* is a bijective, differentiable mapping $f : \mathbb{R}^d \to \mathbb{R}^d$ such that its inverse is differentiable as well.

Theorem 5.8. *The map F_c is a diffeomorphism for all $c > 0$ if and only if*

(i) $\operatorname{sign}(S) \cap \operatorname{sign}(\tilde{S}^\perp) = \{\mathbf{0}\}$,

(ii) *for every non-zero $\tilde{\tau} \in \operatorname{sign}(\tilde{S}^\perp)_\oplus$, there is a non-zero $\tau \in \operatorname{sign}(S^\perp)_\oplus$ such that $\tau \preceq \tilde{\tau}$, and*

(iii) *the pair $(S, \tilde{S})$ is non-degenerate.*

Condition (i) is the same as condition (2) from Corollary 5.3. Therefore, F_c is injective if this condition holds.
In condition (ii), it suffices to consider the cocircuits. Assume that for every positive cocircuit $\tilde{\sigma} \in \operatorname{sign}(\tilde{S}^\perp)$, there is a positive cocircuit $\sigma \in \operatorname{sign}(S^\perp)$ such that $\sigma \preceq \tilde{\sigma}$. Now, let $\tilde{\tau} \in \operatorname{sign}(\tilde{S}^\perp)_\oplus$. By Lemma 4.13, there is a positive cocircuit $\tilde{\sigma} \in \operatorname{sign}(\tilde{S}^\perp)$ such that $\tilde{\sigma} \preceq \tilde{\tau}$. By the assumption, there is a positive cocircuit $\tau \in \operatorname{sign}(S^\perp)$ with $\tau \preceq \tilde{\sigma} \preceq \tilde{\tau}$. Hence, condition (ii) follows.
To check condition (iii), we need to check both conditions of Definition 5.7. The details for checking these conditions will be discussed in the next section.

5.3.1 Non-degenerate

In this section, we discuss Definition 5.7 in more detail. First, note that this definition is a reformulation of the following definition in [MHR19].

Definition 5.9. Let $S, \tilde{S}$ be subspaces of $\mathbb{R}^n$. The pair $(S, \tilde{S})$ is called *non-degenerate* if, for every $z \in \tilde{S}^\perp$ with a positive component,

(I) there is $I = \{i \mid z_i = \lambda\}$ with $\lambda > 0$, defining $\pi \in \{0, +\}^n$ with $\pi^+ = I$, such that $\pi \notin \operatorname{sign}(S)_\oplus$; or

(II) for $\tilde{\tau} = \operatorname{sign}(z) \in \operatorname{sign}(\tilde{S}^\perp)$, there is a non-zero $\tau \in \operatorname{sign}(S^\perp)_\oplus$ such that $\tilde{\tau}^0 \subseteq \tau^0$.

Note that $I = \pi(z_i, z)^+$ and $\pi(z_i, z) = \pi(\lambda, z) = \pi$. Therefore, (I) is equivalent to (i) of Definition 5.7.
In (II), we have to check for each sign vector $\tilde{\tau} \in \operatorname{sign}(\tilde{S}^\perp)$ with a positive component whether there exists a non-zero $\tau \in \operatorname{sign}(S^\perp)_\oplus$ such that $\tilde{\tau}^0 \subseteq \tau^0$. If this is the case, there also exists a sign vector $\sigma \in \operatorname{sign}(S^\perp)_\oplus$ with minimal support such that $\tilde{\tau}^0 \subseteq \tau^0 \subseteq \sigma^0$. Therefore, it suffices to check existence only for the positive cocircuits of $\operatorname{sign}(S^\perp)$.
Furthermore, let $\tilde{\tau} \in \operatorname{sign}(\tilde{S}^\perp)$ be a sign vector with a positive component that satisfies the condition. Then, there is also a cocircuit $\tilde{\sigma} \in \operatorname{sign}(\tilde{S}^\perp)$ with $\tilde{\tau}^0 \subseteq \tilde{\sigma}^0$ that satisfies the condition. Thus, if the condition holds for every positive cocircuit $\tilde{\sigma} \in \operatorname{sign}(\tilde{S}^\perp)$, then it equivalently holds for every positive covector $\tilde{\tau} \in \operatorname{sign}(\tilde{S}^\perp)$. Hence, we only need to take positive cocircuits $\tilde{\sigma} \in \operatorname{sign}(\tilde{S}^\perp)$ with a positive component into account. Since $\tilde{\sigma}^0 = (-\tilde{\sigma})^0$, we can also neglect the condition on the positive component and obtain condition (i) of Definition 5.7.

In the following, we want to check algorithmically whether a pair of subspaces is non-degenerate. It is straightforward to check condition (ii) of Definition 5.7 algorithmically. In contrast, checking condition (i) is more involved. This is because it is not clear how we can compute each relevant vector of $\tilde{S}^\perp$. Instead, we will specify a recursive algorithm and make use of Minty's Lemma. For this purpose, we first negate condition (i).

¬(i) There is $z \in \tilde{S}^\perp$ such that

- for some $i \in \{1, \dots, n\}$, we have $z_i > 0$; and
- for all $i \in \{1, \dots, n\}$ with $z_i > 0$, we have $\pi(z_i, z) \in \operatorname{sign}(S)_\oplus$.

The resulting algorithm attempts to constructs such a z recursively. Here, we will go over the positive covectors in $\operatorname{sign}(S)$ and check with Minty's Lemma, whether a corresponding $z \in \tilde{S}^\perp$ exists. Now, we state this algorithm.

Algorithm 1: `nondeg_init`

Input : matrices W, $\tilde{W}$
Output: boolean
1 compute the set of positive covectors P of $\ker(W)$;
2 define the vector space $\mathcal{V} := \operatorname{row}(\tilde{W})$;
3 val := True;
4 `nondeg_rec`($\mathcal{V}$, P, $\emptyset$);
5 **return** val;

Algorithm 1 is the initial call of the recursive algorithm. The vector space in line 2 is determined by the matrix $\tilde{W}$. During the recursive call, this matrix will be modified. In line 3, we specify a global variable that might be changed to `False` by a recursive call. In this case, we have found a vector that satisfies condition ¬(i) and we return `False`.

Algorithm 2: `nondeg_rec`

Input : vector space $\mathcal{V}$, set of sign vectors P, set of indices J
Output:

```
1  while P ≠ ∅ do
2      X := pop(P);
3      if X⁺ ∩ J = ∅ then
4          define V̂ as the subspace of V where each vector has equal components on X⁺;
5          Ĵ := J ∪ X⁺;
6          if there exists a vector v ∈ V̂ with v_e > 0 for e ∈ Ĵ and v_e ≤ 0 else then
7              val := False;
8              return;
9          else if there exists a vector v ∈ V̂ such that v_e > 0 for e ∈ J then
10             nondeg_rec(V̂, copy(P), Ĵ);
11     if val = False then return;
12 return;
```

To define the subspace $\hat{\mathcal{V}}$ in line 4 of Algorithm 2, we modify the matrix corresponding to $\mathcal{V}$. For each $i \in X^+$ with $i \neq \min(X^+)$, we define the row a^i of a matrix A, where

$$a^i_k := \begin{cases} 1 & \text{if } k = \min(X^+), \\ -1 & \text{if } k = i, \\ 0 & \text{otherwise.} \end{cases}$$

The matrix A consists of $|X^+| - 1$ rows. Note that for $|X^+| = 1$, A is the empty matrix. Now, $v \in \ker(A)$ if and only if v has equal components on X^+. Hence, we consider the subspace $\hat{\mathcal{V}} := \mathcal{V} \cap \ker(A)$.

To check the condition in line 6, we apply Minty's Lemma. Here, we compute the elementary vectors of $\hat{\mathcal{V}}^\perp$. Next, we define the intervals

$$I_k := \begin{cases} (0, \infty) & \text{if } k \in \hat{J}, \\ (-\infty, 0] & \text{else.} \end{cases}$$

In this case, we have found a vector that certifies that condition ¬(i) holds. Therefore, we terminate the algorithm by changing the truth value of the global variable in 7.
Similarly, we check the condition in line 9 by considering the intervals

$$I_k := \begin{cases} (0, \infty) & \text{if } k \in \hat{J}, \\ \mathbb{R} & \text{else.} \end{cases}$$

Note that we can use the elementary vectors computed in line 6.

Omitting line 11 will not change the output of the algorithm. However, the algorithm would continue the loops in the open branches although the result is already `False`. Consequently, line 11 makes the algorithm more efficient by closing those branches immediately.

If condition (i) is not fulfilled, Algorithm 1 will construct a non-empty set of sign vectors $T \subseteq \operatorname{sign}(S)_\oplus$ with pairwise disjoint support. In this case, there is a vector $z \in \tilde{S}^\perp$ with a positive component such that for every $z_i > 0$, we have $\pi(z_i, z) \in T$. Therefore, condition ¬(i) is satisfied.

Conversely, if condition (i) is not satisfied, then Algorithm 1 will find several sets of sign vectors $T \subseteq \operatorname{sign}(S)_{\oplus}$ with pairwise disjoint support such that the condition in line 6 is not satisfied for the corresponding subspace $\hat{\mathcal{V}}$. Therefore, Algorithm 2 will never reach line 7 and consequently Algorithm 1 returns `True`.

5.3.2 Examples

We conclude this section with some examples.

Example 5.10. Let us consider the following matrices:

```
sage: W = matrix([[1,0,1,0],[0,1,0,1]])
sage: W
[1 0 1 0]
[0 1 0 1]
sage: Wt = matrix([[1,0,0,-1],[0,1,1,1]])
sage: Wt
[ 1  0  0 -1]
[ 0  1  1  1]
```

Hence, we obtain the oriented matroids:

```
sage: covectors_from_matrix(W, kernel=True, algorithm='fe', separate=True)
[[(0000)], [(0+0-), (-0+0), (+0-0), (0-0+)], [(-++-), (++--), (+--+),
   (--++)]]
sage: covectors_from_matrix(Wt, algorithm='fe', separate=True)
[[(0000)], [(+00-), (+++0), (-00+), (---0), (0---), (0+++)], [(+++-),
   (---+), (----), (+---), (++++), (-+++)]]
```

We use the method from before to check statement (i):

```
sage: cond_inj_intersection(W, Wt)
True
```

Therefore, the map F_c is injective for all $c > 0$. To check, whether statement (ii) holds, we consider the cocircuits.

```
sage: cc1 = cocircuits_from_matrix(W, kernel=False)
sage: cc1
[(+0+0), (0+0+), (-0-0), (0-0-)]
sage: cc2 = cocircuits_from_matrix(Wt, kernel=False)
sage: cc2
[(+++0), (-00+), (0+++), (---0), (+00-), (0---)]
```

Here, we are only interested in the positive cocircuits.

```
sage: cc1p = [X for X in cc1 if X > 0]
sage: cc1p
[(+0+0), (0+0+)]
sage: cc2p = [X for X in cc2 if X > 0]
sage: cc2p
[(+++0), (0+++)]
```

From the output, we see already that statement (ii) is fulfilled. This is because every element in

`cc2p` has a smaller element in `cc1p`. There is also a function in the package that can be used directly to check whether condition (ii) is fulfilled.

```
sage: cond_faces(W, Wt)
True
```

It remains to check condition (iii). For this purpose, we consider again the oriented matroid $\operatorname{sign}(S)$.

```
sage: covectors_from_matrix(W, kernel=True)
[(0000), (-0+0), (0-0+), (+0-0), (0+0-), (-++-), (++--), (+--+), (--++)]
```

Therefore, $\operatorname{sign}(S)_{\oplus} = \{\mathbf{0}\}$ and consequently the first condition of Definition 5.7 is satisfied for each vector $z \in \tilde{S}^{\perp}$. The package offers a function to check whether condition (iii) is fulfilled. This function consists of two parts. One part is an implementation of Algorithm 1 and the other part checks condition (ii) of Definition 5.7.

```
sage: cond_nondegenerate(W, Wt)
True
```

Hence, the pair $(S, \tilde{S})$ is non-degenerate and thus by Theorem 5.8 the map F_c defined by W and $\tilde{W}$ is a diffeomorphism for all $c > 0$.

Example 5.11. Now, we swap the matrices from Example 5.10.

```
sage: W = matrix([[1,0,0,-1],[0,1,1,1]])
sage: W
[ 1  0  0 -1]
[ 0  1  1  1]
sage: Wt = matrix([[1,0,1,0],[0,1,0,1]])
sage: Wt
[1 0 1 0]
[0 1 0 1]
```

Because of the symmetry of condition (i), we obtain the same result.

```
sage: cond_inj_intersection(W, Wt)
True
```

Therefore, the intersection consists only of the zero sign vector and thus condition (i) of Theorem 5.8 is satisfied. Now, we attempt to check condition (ii).

```
sage: cc1 = cocircuits_from_matrix(W, kernel=False)
sage: cc1
[(+++0), (-00+), (0+++), (---0), (+00-), (0---)]
sage: cc2 = cocircuits_from_matrix(Wt, kernel=False)
sage: cc2
[(+0+0), (0+0+), (-0-0), (0-0-)]
```

Again, we are only interested in the positive cocircuits:

```
sage: cc1p = [X for X in cc1 if X > 0]
sage: cc1p
[(+++0), (0+++)]
sage: cc2p = [X for X in cc2 if X > 0]
```

```
sage: cc2p
[(+0+0), (0+0+)]
```

Therefore, condition (ii) does not hold. We also apply the corresponding function from the package.

```
sage: cond_faces(W, Wt)
False
```

Thus, the map F_c is not a diffeomorphism for all $c > 0$.

Example 5.12. Now, we consider Example 20 from [MHR19]. Here, we have a parameter $\tilde{w} > 0$. Depending on this parameter, the resulting map F_c will be bijective.

```
sage: var('wt')
wt
sage: W = matrix(3, 6, [0,0,1,1,-1,0,1,-1,0,0,0,-1,0,0,1,-1,0,0])
sage: W
[ 0  0  1  1 -1  0]
[ 1 -1  0  0  0 -1]
[ 0  0  1 -1  0  0]
sage: Wt = matrix(3, 6, [1,1,0,0,-1,wt,1,-1,0,0,0,0,0,0,1,-1,0,0])
sage: Wt
[ 1  1  0  0 -1 wt]
[ 1 -1  0  0  0  0]
[ 0  0  1 -1  0  0]
```

The conditions (i) and (ii) depend on the sign vectors of the corresponding oriented matroids S and $\tilde{S}$. Consequently, the choice of the positive parameter $\tilde{w}$ does not affect the result. In order to compute the sign vectors, we set $\tilde{w}$ to 1.

```
sage: cond_inj_intersection(W, Wt(wt=1))
True
```

Hence, condition (i) is satisfied.

```
sage: cond_faces(W, Wt(wt=1))
True
```

Furthermore, condition (ii) holds. For specific values of $\tilde{w}$, the pair $(S, \tilde{S})$ is non-degenerate. This is the case for $\tilde{w} \in (0, 1) \cup (1, 2)$.

```
sage: cond_nondegenerate(W, Wt(wt=1/2))
True
```

```
sage: cond_nondegenerate(W, Wt(wt=3/2))
True
```

On the other hand, condition (iii) does not hold if $\tilde{w} \in \{1\} \cup [2, \infty)$. In this case, the map F_c is injective but not surjective.

```
sage: cond_nondegenerate(W, Wt(wt=1))
False
```

```
sage: cond_nondegenerate(W, Wt(wt=2))
False

sage: cond_nondegenerate(W, Wt(wt=3))
False
```

Consequently, the map F_c is a diffeomorphism if $\tilde{w} \in (0,1) \cup (1,2)$.

The package also offers a function that returns a certificate that condition (i) of Definition 5.7 is violated. Here, we include the respective function from the package.

```
sage: from bijectivity_exponential_maps.conditions_bijectivity import
    nondeg_cond1
```

Example 5.13. We consider the following matrices:

```
sage: W = matrix([[1,1,0,0],[0,0,1,0]]).right_kernel_matrix()
sage: Wt = matrix([[1,0,2,0],[0,1,0,-1]])
```

Now, we check condition (i).

```
sage: nondeg_cond1(W, Wt, certificate=True)
[False, [[[2], [0, 1]], (2, 2, 4, -2)]]
```

From the output, we see that the condition is violated. The vector $(2,2,4,-2)$ lies in the row space of $\tilde{W}$ and corresponds to the positive sign vectors

$$(\,0\;\;0\;+\;0\,) \quad \text{and} \quad (\,+\;+\;0\;\;0\,)$$

in $\operatorname{sign}(S)_\oplus$ that are represented by the sets $\{2\}$ and $\{0,1\}$.

5.4 Characterizing robust bijectivity

Now, we study small perturbations of the exponents $\tilde{W}$ such that the signs of the non-zero maximal minors of $\tilde{W}$ do not change. This corresponds to small perturbations of the subspace $\tilde{S}$ (in the Grassmannian) and will lead to a new subspace $\tilde{S}_\varepsilon$. For further details and references, consult [MHR19].

The next result is Corollary 33 from [MHR19].

Corollary 5.14. *The following statements are equivalent:*

(1) *F_c is a diffeomorphism for all $c > 0$ and all small perturbations $\tilde{S}_\varepsilon$.*

(2) $\operatorname{sign}(S) \subseteq \overline{\operatorname{sign}(\tilde{S})}$

(3) $\det(W_I) \neq 0$ *implies* $\det(W_I)\det(\tilde{W}_I) > 0$ *for all subsets $I \subseteq \{1,\dots,n\}$ of cardinality d (or "< 0" for all I).*

To check condition (2), we do not need to compute the closure. It suffices to check whether for each tope $\tau \in \operatorname{sign}(S)$ there exists a tope $\tilde{\tau} \in \operatorname{sign}(\tilde{S})$ with $\tau \preceq \tilde{\tau}$. Now, let $\sigma \in \operatorname{sign}(S)$. Then, there exists a tope $\tau \in \operatorname{sign}(S)$ with $\sigma \preceq \tau$ and by assumption $\sigma \preceq \tau \preceq \tilde{\tau}$. In this case, we also have $\sigma \in \overline{\operatorname{sign}(\tilde{S})}$ and condition (2) follows.

5.4.1 Examples

Now, we check the conditions of Corollary 5.14 for several examples.

Example 5.15. Let us consider the following matrices:

```
sage: W = matrix([[1,0,-1,0],[0,1,0,0]])
sage: W
[ 1  0 -1  0]
[ 0  1  0  0]
sage: Wt = matrix([[1,0,-1,0],[0,1,-1,1]])
sage: Wt
[ 1  0 -1  0]
[ 0  1 -1  1]
```

In order to check condition (2) of Corollary 5.14, we consider the topes of the corresponding oriented matroids.

```
sage: topes_from_matrix(W, kernel=True)
[(+0+-), (-0--), (-0-+), (+0++)]
sage: topes_from_matrix(Wt, kernel=True)
[(++++), (+-++), (---+), (+++-), (-+--), (----)]
```

One can already see that for every tope X of the oriented matroid corresponding to W there is a tope Y corresponding to $\tilde{W}$ such that $X \preceq Y$. Hence, statement (2) is fulfilled.
The package [bijectivity_exponential_maps] also offers a function that checks this condition directly.

```
sage: cond_closure_sign_vectors(W, Wt)
True
```

Now, we compute the maximal minors of the two matrices to check whether condition (3) is satisfied.

```
sage: W.minors(2)
[1, 0, 0, 1, 0, 0]
sage: Wt.minors(2)
[1, -1, 1, 1, 0, -1]
```

From the output, we see whenever a minor of W is non-zero. Then, the corresponding minor of $\tilde{W}$ has the same sign. Hence, condition (3) is fulfilled. This condition is also implemented in the package.

```
sage: cond_closure_minors(W, Wt)
True
```

Thus, by Corollary 5.14, the map F_c is a diffeomorphism for all $c > 0$ and all small perturbations $\tilde{S}_\varepsilon$.

Example 5.16. Now, consider the following matrices:

```
sage: var('a,b,c')
(a, b, c)
sage: W = matrix([[1,0,-1],[0,c,-1]])
sage: W
```

```
[ 1  0 -1]
[ 0  c -1]
sage: Wt = matrix([[1,0,a],[0,1,b]])
sage: Wt
[1 0 a]
[0 1 b]
```

We cannot check condition (2) of Corollary 5.14 since there are variables in W and $\tilde{W}$. Therefore, we want to obtain equations on the variables a, b, c such that condition (3) is satisfied. First, we compute the minors of the matrices:

```
sage: W.minors(2)
[c, -1, c]
sage: Wt.minors(2)
[1, b, -a]
```

The corresponding function from the package supports symbolic matrices as input. In this case, we obtain the following equations on the variables.

```
sage: cond_closure_minors(W, Wt)
[[-b > 0, [c == 0, 'or', c > 0], [c == 0, 'or', -a*c > 0]], 'or', [-b <
   0, [c == 0, 'or', c < 0], [c == 0, 'or', -a*c < 0]]]
```

From this system of equations, we can induce that $b \neq 0$ in order to satisfy this condition. Indeed, if $b = 0$, we obtain:

```
sage: cond_closure_minors(W, Wt(b=0))
False
```

Therefore, assume for instance $b = 2$.

```
sage: cond_closure_minors(W, Wt(b=2))
[[c == 0, 'or', c < 0], [c == 0, 'or', -a*c < 0]]
```

From the output, we see that either $c = 0$ or $c < 0$. Otherwise, we obtain:

```
sage: cond_closure_minors(W(c=1), Wt(b=2))
False
```

If $c = 0$, then the result will be `True`.

```
sage: cond_closure_minors(W(c=0), Wt(b=2))
True
```

For $c < 0$, we are left with a condition on a.

```
sage: cond_closure_minors(W(c=-1), Wt(b=2))
[a < 0]
```

Analogously, we can assume that $b < 0$ and we will obtain similar conditions on the other variables.

A Computing elementary vectors over integral domains

In this section, we will investigate the construction of all elementary vectors lying in the kernel of a given matrix. The naive way is to set up and solve a linear system for each elementary vector. Instead, we compute all maximal minors of the matrix and apply Cramer's rule to construct each elementary vector. This will lead us to an algorithm that is implemented in the package [`elementary_vectors`]. For further details on the algorithm consult [Aic20].

Many of the results in this chapter work for commutative rings. However, for simplicity, we will state the results for integral domains. In this chapter, let $\mathcal{R}$ denote an integral domain.
First, we need to consider matrices that are restricted to certain components.

Definition A.1. For a set $I \subseteq \{1, \dots, n\}$, we denote by $M_I \in \mathcal{R}^{r \times |I|}$ the submatrix consisting of the columns M_i with $i \in I$ such that the column indices are in ascending order.

Now, we will introduce a version of Cramer's Rule that can be applied to matrices of commutative rings.

Lemma A.2. *Let $A \in \mathcal{R}^{n \times n}$ be a matrix and let $b \in \mathcal{R}^n$ be a vector such that there exists $x \in \mathcal{R}^n$ with $Ax = b$. Then, for each $k = 1, \dots n$ we have*

$$\det(A)x_k = \det(A^{(k)}),$$

where $A^{(k)}$ is the matrix we obtain by exchanging the k-th column of A by b.

Maximal minors play an important role for constructing elementary vectors.

Definition A.3. Let $M \in \mathcal{R}^{r \times n}$ and $J \subseteq \{1, \dots, n\}$ with $|J| = r$. By

$$M(J) := \det(M_J),$$

we denote the *maximal minor* of M corresponding to J.

Roughly speaking, the function pos gives us the position of a component in a set sorted in ascending order.

Definition A.4. Let $I \subseteq \{1, \dots, n\}$ be a non-empty set. We define the function

$$\begin{aligned} \operatorname{pos}\colon \mathcal{P}(\{1, \dots, n\}) \times I &\to \mathbb{N}, \\ (I, k) &\mapsto |\{i \in I : \ i < k\}|. \end{aligned}$$

The next theorem gives us an algorithm. First, we compute all maximal minors of a matrix M of rank r. Then, we construct for each submatrix of M consisting of $r+1$ columns a vector using the precomputed maximal minors. The constructed vectors will be elementary vectors in the kernel of M if the corresponding submatrices have maximal rank.

Theorem A.5. *Let $M \in \mathcal{R}^{r \times n}$ and let $I \subseteq \{1, \dots, n\}$ with $|I| = r + 1$ and $\operatorname{rank} M_I = r$. Define the vector $v \in \mathcal{R}^n$ by*

$$v_k := \begin{cases} (-1)^{\operatorname{pos}(I,k)} M(I \setminus \{k\}) & \text{if } k \in I, \\ 0 & \text{else.} \end{cases}$$

Then, $v \in \ker(M)$ and v is an elementary vector corresponding to I, that is, $\underline{v} \subseteq I$.

Theorem A.5 can be proven with help of Lemma A.2. The proof is presented in [Aic20].

B Lattice theory

Lattices play an important role for oriented matroids. In this appendix, we recall basic notions of partially ordered sets and lattices, see for example [Grä78].

B.1 Partially ordered sets

First, we recall several properties of binary relations.

Definition B.1. Let $\leq$ be a binary relation on a set P. For all $a, b, c \in P$, we have

(P1) (reflexivity) $a \leq a$;

(P2) (transitivity) if $a \leq b$ and $b \leq c$, then $a \leq c$;

(P3) (antisymmetry) if $a \leq b$ and $b \leq a$, then $a = b$;

(P4) (linearity) $a \leq b$ or $b \leq a$.

A *partially ordered set* or *poset* is a set P with a binary relation $\leq$ such that for all $a, b, c \in P$ the axioms (P1), (P2) and (P3) hold.

In the following, let $(P, \leq)$ be a poset.

Definition B.2. Let $a, b \in P$. If $a \leq b$ or $b \leq a$, we say that a and b are *comparable*.

Definition B.3. Let $a, b \in P$. We define $a < b$ if $a \leq b$ and $a \neq b$.
An element $a \in P$ is *covered* by $b \in P$ or $b \in P$ *covers* $a \in P$ if $a < b$ and there is no $c \in P$ such that $a < c < b$.

Definition B.4. Let $a \in P$. If for every $b \in P$ we have $b \leq a$ implies $b = a$, then a is called a *minimal* element of P. By

$$\min(P) := \{a \in P \mid a \text{ is a minimal element of } P\},$$

we denote the set of minimal elements of P.

Let $a \in P$. If for every $b \in P$ we have $a \leq b$ implies $b = a$, then a is called a *maximal* element of P. By

$$\max(P) := \{a \in P \mid a \text{ is a maximal element of } P\},$$

we denote the set of maximal elements of P.

Definition B.5. An element $b \in P$ is called *bottom element* of P if $b \leq a$ for every $a \in P$. An element $t \in P$ is called *top element* of P if $a \leq t$ for every $a \in P$.

Next, we define a special poset, where every two elements are comparable.

Definition B.6. Let $(P, \leq)$ be a poset that satisfies (P4). Then, P is called a *chain*.
Let $(P, \leq)$ be a poset and let $\emptyset \neq C \subseteq P$. We call C a *chain* of P if C is a chain as a subposet. The length $l(C)$ of a finite chain is $|C| - 1$. A poset P has the length $n \in \mathbb{N}$ if there is a chain in P with length n and all chains in P have at most length n.
Let C be a finite chain in P. If for every finite chain $\tilde{C}$ in P with the same bottom and top element we have $l(\tilde{C}) \leq l(C)$, then C is a maximal chain in P.

B.2 Lattices

Lattices are a special kind of posets.

Definition B.7. Let $(P, \leq)$ be a poset and let $a, b \in P$. An element $x \in P$ is called a *lower bound* of a and b if

$$x \leq a \quad \text{and} \quad x \leq b.$$

Similarly, we call an element $x \in P$ an *upper bound* of a and b if

$$a \leq x \quad \text{and} \quad b \leq x.$$

If the set of lower bounds of a and b has a largest element, we call it *infimum* of a and b and denote it by $\inf(a, b)$.
Similarly, if the set of upper bounds of a and b has a smallest element, we call it *supremum* of a and b and denote it by $\sup(a, b)$.

A lattice is a poset, where every two elements have an infimum and a supremum.

Definition B.8. Let $(P, \leq)$ be a poset Then, P is called a *lattice* if every two elements $a, b \in P$ have an infimum $\inf(a, b) \in P$ and a supremum $\sup(a, b) \in P$.

C Implementation

In this section, we list documentation of functions from the packages presented in this thesis.

C.1 Elementary vectors

This section covers functions from the package [elementary_vectors].

```
sage: elementary_vectors.__doc__
```

```
Computes elementary vectors of a subspace determined by the matrix "M".

INPUT:

- "M" -- a matrix

- "kernel" -- a boolean (default:  "False")

OUTPUT:

- If "kernel" is false, returns a list of elementary vectors lying in
  the row space of "M".  (default)

- If "kernel" is true, returns a list of elementary vectors lying in
  the kernel of "M".
```

```
sage: conformal_elimination.__doc__
```

```
Applies conformal elimination to two real vectors to find a new vector.

INPUT:

- "x" -- a real vector

- "y" -- a real vector

- "S" -- a list of indices (default:  "[]")

OUTPUT:

Returns a new vector "z = x + a y" where "a > 0", such that "z[e] == 0"
for some "e" in "S" and "Z_S <= X_S" and "Z_f = (X o Y)_f" for "f"
not in "D(X, Y)".  Here, "X", "Y" and "Z" are the sign vectors
corresponding to "x", "y" and "z".

..  NOTE::

    If "S" is the empty list "[]", the whole list of separating elements
    will be considered instead.  (default)
```

The function `exists_vector` is an implementation of Minty's Lemma.

```
sage: exists_vector.__doc__

 Returns whether a vector exists in the vector space determined by a matrix
 such that the components lie in given intervals.

 INPUT:

 - "data" -- either a real matrix with "n" columns or a list of
             elementary vectors of length "n"

 - "L" -- a list of real values ("-oo" and "oo" are accepted) of length "n"

 - "R" -- a list of real values ("-oo" and "oo" are accepted) of length "n"

 - "l" -- a boolean (default:  "True") or a list of booleans of length "n"

 - "r" -- a boolean (default:  "True") or a list of booleans of length "n"

 - "kernel" -- a boolean (default:  "False")

 - "certificate" -- a boolean (default:  "False")

 OUTPUT:

 - If "data" is a matrix, then the elementary vectors of "M" are computed.

     - If "kernel" is false, considers the vector space generated by the
       rows of the matrix "M".  (default)
       In this case, the elementary vectors will lie in the kernel of "M".

     - If "kernel" is true, considers the vector space generated by the
       kernel of the matrix "M".
       In this case, the elementary vectors will lie in the row space of "M".

 - If "data" is a list of elementary vectors, then those will be used.

     - In this case, the argument "kernel" will be ignored.

 - "L" and "R" are the left and right interval values, respectively.

 - "l" and "r" determine the intervals.

 - The left (or right) interval half of the "i"-th interval is

     - closed if "l[i]" (or "r[i]") is "True" (default).

     - open if "l[i]" (or "r[i]") is "False".

 - If "l" (or "r") is a boolean, then all left (or right) interval halves
   are considered closed if "True" (default) and open if "False".

 - If "certificate" is "False" (default), returns a boolean.

 - If "certificate" is "True" and the result is false, then a list
   "[False, v]" will be returned.  Here, "v" is an elementary vector of "M"
   that certifies that there exists no vector.
```

C.2 Sign vectors

The package [sign_vectors] contains a class for sign vectors, functions to construct sign vectors and several operations that can be applied to sets of sign vectors. Moreover, this package includes a submodule to work with oriented matroids.

C.2.1 Class `SignVector`

Here, we list the most important methods of the class `SignVector`.

```
sage: SignVector.__doc__
A sign vector is an element of "{-,+,0}^n".
sage: SignVector.compose.__doc__

 Returns the composition of two sign vectors.

 INPUT:

 - "other" -- a sign vector

 OUTPUT:

 Composition of this sign vector with "other".

 ..  NOTE::

     Alternatively, the operator "&" can be used.

sage: SignVector.support.__doc__
Returns a list of indices where the sign vector is non-zero.
sage: SignVector.zero_support.__doc__
Returns a list of indices where the sign vector is zero.
sage: SignVector.positive_support.__doc__
Returns a list of indices where the sign vector is positive.
sage: SignVector.negative_support.__doc__
Returns a list of indices where the sign vector is negative.
sage: SignVector.list_from_positions.__doc__
Returns a list of the entries in the list "S".
sage: SignVector.separating_elements.__doc__

 Computes the list of separating elements of two sign vectors.

 INPUT:

 - "other" -- sign vector

 OUTPUT:
 List of elements "e" such that "self[e] == -other[e] != 0".
```

```
sage: SignVector.conforms.__doc__
Conformal relation of two sign vectors.
sage: SignVector.is_orthogonal_to.__doc__

 Returns whether two sign vectors are orthogonal.

 INPUT:

 - "other" -- a sign vector.

 OUTPUT:

 - Returns "True" if the sign vectors are orthogonal and "False" otherwise.
```

C.2.2 Functions for generating sign vectors

The package [sign_vectors] offers several functions for creating sign vectors.

```
sage: sign_vector.__doc__

 Create a sign vector from a list, vector or string.

 INPUT:

 - "v" -- different inputs are accepted:

     - an iterable (e.g.  a list or vector) of real values.

     - a string consisting of '-', '+', '0'.  Other characters are treated as '0'.

 EXAMPLES::

   sage:  from sign_vectors import sign_vector
   sage:  sign_vector([5,0,-1,-2])
   (+0--)
   sage:  sign_vector('++-+-00-')
   (++-+-00-)]

sage: zero_sign_vector.__doc__
Return the zero sign vector of length "n".
sage: random_sign_vector.__doc__
Return a random sign vector of length "n".
```

C.2.3 Functions for lists of sign vectors

Here are some functions that can be applied to lists of sign vectors.

```
sage: closure.__doc__

 Computes the closure of a list of sign vectors.

 INPUT:

 - "W" -- a list of sign vectors

 - "separate" -- boolean (default:  "False")

 OUTPUT:

 If "separate" is false, return the closure of "W".  (default)

 If "separate" is true, separate the closure into lists, where each element
 has the same number of zero entries.

sage: deletion.__doc__

 Removes the components corresponding to "R" from a list of sign vectors.
 Also works for real vectors.

 INPUT:

 - "F" -- a list of sign vectors, a list of real vectors, or a matrix.

 - "R" -- a list of indices.

sage: contraction.__doc__

 Returns all sign vectors that are zero on "R".  Also works for real vectors.

 INPUT:

 - "F" -- a list of sign vectors, a list of vectors, or a matrix.

 - "R" -- a list of indices.

 - "keep_components" -- a boolean (default:  "False").

 OUTPUT:

 - If "keep_components" is false, remove entries in "R".  (default)

 - If "keep_components" is true, keep entries in "R".
```

Next, we list the documentation of some utility functions.

```
sage: loops.__doc__

 Computes the list of loops of a given list of sign vectors "W".

 ..NOTE::

     A loop is a component where every sign vector of "W" is zero.
```

```
sage: parallel_classes.__doc__

 Computes the parallel classes of a given set of sign vectors "W".

 INPUT:

 - "W" -- a list of sign vectors or vectors of length "n".

 OUTPUT:

 - a partition of "[0, ..., n-1]" into parallel classes.
```

C.2.4 Oriented matroids

Oriented matroids can be represented by a list of sign vectors.

The function `cocircuits_from_matrix` computes a list of elementary vectors corresponding to a matrix. Then, it applies the sign function to those elementary vectors and the elementary vectors multiplied by -1. That way, we obtain all cocircuits of the oriented matroid corresponding to the given matrix.

```
sage: cocircuits_from_matrix.__doc__

 Computes a list of cocircuits determined by the matrix "A".

 INPUT:

 - "A" -- a matrix with real arguments.

 - "kernel" -- a boolean (default:  False)

 OUTPUT:

 - If "kernel" is false, returns a list of cocircuits determined by the row
   space of the matrix "A" (default).

 - If "kernel" is true, returns a list of cocircuits determined by the
   kernel of the matrix "A".
```

To compute all covectors, we can compose cocircuits in a structured way. A pseudocode of the corresponding algorithm can be found in [Fin01].

```
sage: covectors_from_cocircuits.__doc__

 Uses a list of cocircuits to compute all covectors of the corresponding
 oriented matroid.

 INPUT:

 - "L" -- a list of cocircuits of an oriented matroid.

 OUTPUT:

 - a list of all covectors of the oriented matroid.
```

We compute all topes, by composing cocircuits. For a pseudocode of the resulting algorithm, see [Fin01].

```
sage: topes_from_cocircuits.__doc__
```

```
Uses the cocircuits of an oriented matroid to compute the topes.

INPUT:

- "D" -- a list of cocircuits of an oriented matroid.

OUTPUT:

- a list of topes of the oriented matroid.
```

Given a list of all covectors with the same rank, we compute the covectors with one less rank. The corresponding pseudocode is available in [Fin01].

```
sage: lower_faces.__doc__
```

```
Computes a list of lower faces.

INPUT:

- "W" -- a list of all covectors with same rank "r" of an oriented matroid.

OUTPUT:

Returns a list of covectors of rank "r-1" of the oriented matroid.
```

The function `face_enumeration` applies `lower faces` several times to reach all covectors with smaller rank than the given covectors. For a pseudocode, consult [FST91] or [Fin01].

```
sage: face_enumeration.__doc__
```

```
Computes all covectors with less rank than the given list of covectors.

INPUT:

- "W" -- a list of all covectors of same rank "r" of an oriented matroid.

OUTPUT:

Returns a list of lists.  Every list consists of all covectors of the same rank
smaller than or equal to "r" of the oriented matroid.
```

```
sage: topes_from_matrix.__doc__

 Returns a list of topes of the oriented matroid corresponding to the matrix "A".

 INPUT:

 - "A" -- a matrix

 - "kernel" -- a boolean (default:  "False")

 OUTPUT:

 - If "kernel" is false, returns a list of topes determined by the row space
   of the matrix "A".  (default)

 - If "kernel" is true, returns a list of topes determined by the kernel of
   the matrix "A".

sage: covectors_from_topes.__doc__

 Computes all covectors from a list of topes.

 INPUT:

 - "T" -- a list of topes.

 - "separate" -- a boolean (default:  "False")

 OUTPUT:
 The list of covectors of the corresponding oriented matroid.

 - If "separate" is false, returns a list of covectors.  The covectors are
   sorted by rank.  (default)

 - If "separate" is true, returns a list of lists of covectors, separated
   by their rank.

sage: cocircuits_from_topes.__doc__

 Computes all cocircuits from a list of topes.

 INPUT:

 - "T" -- a list of topes.

 OUTPUT:
 A list of cocircuits of the corresponding oriented matroid.
```

```
sage: covectors_from_matrix.__doc__

 Returns a list of covectors of the oriented matroid corresponding to the
 matrix "A".

 INPUT:

 - "A" -- a matrix.

 - "kernel" -- a boolean (default:  "False")

 - "algorithm" -- (optional) either 'face_enumeration' or 'fe'

 - "separate" -- a boolean (default:  "False")

 OUTPUT:

 Returns the list of covectors of an oriented matroid corresponding to the
 matrix "A".

 - If "kernel" is false, the returned covectors will be determined by the
   row space of the matrix "A".  (default)

 - If "kernel" is true, the returned covectors will be determined by the
   kernel of the matrix "A".

 - If "algorithm" is 'face_enumeration' or the shortcut 'fe',

   - if "separate" is false, returns a list of covectors.  The covectors are
     sorted by rank (default).

   - if "separate" is true, returns a list of lists of covectors, separated
     by their rank.
```

C.3 Bijectivity of exponential maps

In this section, we list the documentation of the package [bijectivity_exponential_maps].

C.3.1 Functions

Generalized polynomial and exponential maps can be represented by matrices. Such maps can be computed by the corresponding functions of the package.

```
sage: f_pol.__doc__

 Returns the polynomial map determined by the matrices "W" and "Wt".

 INPUT:

 - "W" -- a matrix

 - "Wt" -- a matrix

 - "c" -- a vector (optional)

 OUTPUT:

 - If "c" is omitted, the result take the vector consisting of ones.
```

```
sage: f_exp.__doc__

 Returns the exponential map determined by the matrices "W" and "Wt".

 INPUT:

 - "W" -- a matrix

 - "Wt" -- a matrix

 - "c" -- a vector (optional)

 OUTPUT:

 - If "c" is omitted, the result take the vector consisting of ones.
```

C.3.2 Conditions

Several conditions for exponential maps can be checked algorithmically.

```
sage: cond_inj_intersection.__doc__

 Returns whether the intersection of the oriented matroids corresponding to
 "W" and "right_kernel(Wt)" consists of the zero sign vector only.

 INPUT:

 - "W" -- a matrix with "n" columns

 - "Wt" -- a matrix with "n" columns

 OUTPUT:
 a boolean
```

```
sage: cond_inj_minors.__doc__

 Returns whether the products of the corresponding maximal minors of the
 matrices "W" and "Wt" have the same sign.

 INPUT:

 - "W" -- a matrix

 - "Wt" -- a matrix with the same dimensions as "W"

 OUTPUT:
 A boolean or a symbolic expression if variables occur.

sage: cond_faces.__doc__

 Returns whether every positive sign vector "X" corresponding to the rows of
 "Wt" has a positive sign vector "Y" corresponding to the rows of "W"
 such that "Y <= X".

 INPUT:

 - "W" -- a matrix with "n" columns

 - "Wt" -- a matrix with "n" columns

 OUTPUT:
 a boolean

sage: nondeg_cond1.__doc__

 Returns whether the first condition of "cond_nondegenerate" is satisfied.

 INPUT:

 - "W" -- a matrix with "n" columns

 - "Wt" -- a matrix with "n" columns

 - "certificate" -- a boolean (default:  "False")

 OUTPUT:
 a boolean

 - If "certificate" is true:

   - If the result is true, returns a vector as a certificate.
```

```
sage: cond_nondegenerate.__doc__

 Let "S", "St" be the sub spaces corresponding to the matrices "W",
 "Wt".  Returns whether the pair "(S, St)" is non-degenerate.

 INPUT:

 - "W" -- a matrix with "n" columns

 - "Wt" -- a matrix with "n" columns

 - "certificate" -- a boolean (default:  "False")

 OUTPUT:
 a boolean

 - If "certificate" is true:

   - If the result is true, returns a vector as a certificate.
```

```
sage: cond_closure_sign_vectors.__doc__

 Returns whether the oriented matroid corresponding to "W" is a subset of
 the closure of the oriented matroid corresponding to "Wt".

 INPUT:

 - "W" -- a matrix with "n" columns

 - "Wt" -- a matrix with "n" columns

 OUTPUT:
 a boolean
```

```
sage: cond_closure_minors.__doc__

 Returns whether for each non-zero maximal minor "m" of "W", we have
 either "m mt > 0" for each corresponding maximal minor "mt" of "Wt" or
 "m mt < 0" for each corresponding maximal minor "mt" of "Wt".

 INPUT:

 - "W" -- a matrix

 - "Wt" -- a matrix with the same dimensions as "W"

 OUTPUT:
 A boolean or a symbolic expression if variables occur.
```

References

[Aic20] Aichmayr, M.: „Computing Elementary Vectors of a linear subspace“. BA thesis. Linz, Austria: Johannes Kepler University Linz, 2020.

[BLS99] Björner, A. et al.: *Oriented Matroids*. 2nd ed. Encyclopedia of Mathematics and its Applications. Cambridge University Press. doi: 10.1017/CBO9780511586507.

[CGPS10] Craciun, G. ; García-Puente, L. D., and Sottile, F.: „Some Geometrical Aspects of Control Points for Toric Patches“. In: *Mathematical Methods for Curves and Surfaces*. Ed. by Dæhlen, M. et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 111–135. isbn: 978-3-642-11620-9.

[Fin01] Finschi, L.: „A graph theoretical approach for reconstruction and generation of oriented matroids“. PhD thesis. Zurich: ETH Zurich, 2001. doi: 10.3929/ethz-a-004255224.

[FST91] Fukuda, K. ; Saito, S., and Tamura, A.: „Combinatorial face enumeration in arrangements and oriented matroids“. In: *Discrete Applied Mathematics* 31.2 (1991), pp. 141–149. doi: 10.1016/0166-218X(91)90066-6.

[Fuk04] Fukuda, K.: *Lecture notes on oriented matroids and geometric computation*. 2004. http://infoscience.epfl.ch/record/77548.

[Grä78] „Chapter I - First Concepts“. In: *General Lattice Theory*. Ed. by Grätzer, G. Vol. 75. Pure and Applied Mathematics. Elsevier, 1978, pp. 1–58. doi: 10.1016/S0079-8169(08)61352-5.

[MHR19] Müller, S. ; Hofbauer, J., and Regensburger, G.: „On the bijectivity of families of exponential/generalized polynomial maps“. In: *SIAM Journal on Applied Algebra and Geometry* 3.3 (2019), pp. 412–438. doi: 10.1137/18M1178153.

[Min74] Minty, G. J.: „A 'from scratch' proof of a theorem of Rockafellar and Fulkerson“. In: *Mathematical Programming* 7 (1974), pp. 368–375.

[MR12] Müller, S. and Regensburger, G.: „Generalized Mass Action Systems: Complex Balancing Equilibria and Sign Vectors of the Stoichiometric and Kinetic-Order Subspaces“. In: *SIAM Journal on Applied Mathematics* 72.6 (2012), pp. 1926–1947. doi: 10.1137/110847056.

[MR16] Müller, S. and Regensburger, G.: „Elementary Vectors and Conformal Sums in Polyhedral Geometry and their Relevance for Metabolic Pathway Analysis“. In: *Frontiers in Genetics* 7.90 (2016).

[Roc69] Rockafellar, R. T.: „The Elementary Vectors of a Subspace of R^N“. In: Combinatorical Mathematics and its Applications (Chapel Hill, University of North Carolina Press), 1969. Chap. 7, pp. 104–127.

[Roc70] Rockafellar, R. T.: *Convex Analysis*. Princeton University, 1970.

[Sage] SageMath: *SageMath*. Version 9.2. 2021. http://www.sagemath.org/.

[Zie95] Ziegler, G. M.: *Lectures on Polytopes*. New York: Springer, 1995.

Packages

[bijectivity_exponential_maps] AICHMAYR, M.: bijectivity_exponential_maps. Version 0.1. 2021. https://github.com/MarcusAichmayr/bijectivity_exponential_maps.

[elementary_vectors] AICHMAYR, M.: elementary_vectors. Version 0.1. 2021. https://github.com/MarcusAichmayr/sign_vectors.

[sign_vectors] AICHMAYR, M.: sign_vectors. Version 0.1. 2021. https://github.com/MarcusAichmayr/sign_vectors.

Index

antisymmetry, 60

big face lattice, 21
bottom element, 60

chain, 60
closure, 14
cocircuit, 22
comparable, 60
composition, 9
cone, 45
conform, $\preceq$, 3, 11
contraction, 35
contraction minor, 35
corresponding oriented matroid, 17
covector, 17
covector axioms, 17
covector elimination, 17
cover, 60

deletion, 34
deletion minor, 34
diffeomorphism, 50
dimension, 33
dual space, 42

elementary vector, 2

face, 17
face of, 11

infimum, 61

Jordan-Dedekind chain property, 33

lattice, 61
linearity, 60
loop, 26

lower bound, 61

maximal, 60
maximal minor, 59
minimal, 60

negative support, 8
non-degenerate, 50, 51

oriented matroid, 17
orthogonal, 13
orthogonal complement, 14
orthogonal space, 42

parallel, 29, 30
parallel class, 29
partially ordered set, 60
poset, 60
positive support, 8

rank, 33
realizable, 17
reflexivity, 60

separate, 10
sign, 3
sign vector, 3, 7
support, 2, 8
support-minimal, 2, 9
supremum, 61

top element, 60
tope, 25
transitivity, 60

upper bound, 61

zero support, 8