mathematics

MDPI

Article

Integration of the Kenzo System within SageMath for New Algebraic Topology Computations

Julián Cuevas-Rozo [1,2], Jose Divasón [1], Miguel Marco-Buzunáriz [3] and Ana Romero [1,*]

[1] Department of Mathematics and Computer Science, University of La Rioja, 26006 Logroño, Spain; jucuevas@unirioja.es (J.C.-R.); jose.divason@unirioja.es (J.D.)
[2] Department of Mathematics, National University of Colombia, Bogotá 111321, Colombia
[3] Department of Mathematics, University of Zaragoza, 50009 Zaragoza, Spain; mmarco@unizar.es
* Correspondence: ana.romero@unirioja.es

Abstract: This work integrates the Kenzo system within Sagemath as an interface and an optional package. Our work makes it possible to communicate both computer algebra programs and it enhances the SageMath system with new capabilities in algebraic topology, such as the computation of homotopy groups and some kind of spectral sequences, dealing in particular with simplicial objects of an infinite nature. The new interface allows computing homotopy groups that were not known before.

Keywords: computer algebra; Kenzo interface; computational algebraic topology; homotopy groups; spectral sequences

Citation: Cuevas-Rozo, J.; Divasón, J.; Marco-Buzunáriz, M.; Romero, A. Integration of the Kenzo System within SageMath for New Algebraic Topology Computations. *Mathematics* **2021**, *9*, 722. https://doi.org/10.3390/math9070722

Academic Editor: Eugenio Roanes-Lozano

Received: 27 February 2021
Accepted: 19 March 2021
Published: 26 March 2021

1. Introduction

Computational algebraic topology is an active area of research in mathematics with numerous applications in different fields, ranging from data analysis [1] to image processing [2]. There exist several programs specialized in algebraic topology computations, such as ChomP [3], Perseus [4], and Javaplex [5]. Such programs are mainly focused on computing topological invariants of finite spaces (by finite, we mean complexes with a finite number of generators in each degree).

Effective homology is a technique developed by Sergeraert [6] which permits carrying out some kinds of computations over infinite structures, via a homotopy equivalence between chain complexes. Kenzo [7] is a computer algebra system that implements several algorithms to compute the homology of infinite structures using effective homology. Infinite spaces can arise in different areas of algebraic topology, for instance when loop spaces or Eilenberg–MacLane spaces are involved. In addition, it also permits one to compute algorithmically homotopy groups combining the Whitehead tower method [8] and the effective homology technique, although it is restricted to 1-reduced simplicial sets. In particular, let us remark that the Whitehead tower method requires the construction of infinite Eilenberg–MacLane spaces. As far as we know, Kenzo is the only program that is able to carry out this kind of computation over infinite structures, which makes it a powerful software. Indeed, Kenzo has obtained some results which had never been determined before [9]. However, it lacks a friendly interface and its use demands a deep knowledge on the Common Lisp programming language, which can be considered tedious. Moreover, the last official Kenzo release is specifically written in Allegro Common Lisp, and it is not fully compatible with other Common Lisp dialects. There exist additional modules for computing other algebraic topology structures such as spectral sequences, pushouts, or discrete vector fields but they are available only in the personal websites of the authors, usually for different outdated Kenzo versions, so that their installation is not simple. Thus, the community around Kenzo is rather small and its features remain unknown for many researchers in algebraic topology.

In order to spread and ease the use of Kenzo, in 2011, a graphical interface for Kenzo called fKenzo was launched [10]. However, it has barely been used due to its limitations: It is too simplistic to deal with the richness of Kenzo and it is not maintained anymore.

That is, it can be stated that there exists no friendly and usable software to carry out powerful computations (in terms of involving infinite structures) in algebraic topology.

On the other hand, SageMath [11] is a general purpose computer algebra system. It is one of the most used programs in both education and research environments, in a wide range of mathematical branches. It was created by William Stein in 2005 with the mission statement of "Creating a viable free open source alternative to Magma, Maple, Mathematica, and Matlab" and since then, it has become a much bigger project, with hundreds of developers from around the world. It follows a "release quickly, release often" approach, resulting in new versions with improvements and bugfixes being published every few months. It uses Jupyter notebooks as a graphical user interface, and it is mainly developed in Python. It follows the philosophy of "building the car instead of reinventing the wheel", that is, it includes many other libraries and systems developed for specific purposes, making them work together under a common abstraction layer. Besides that, a lot of specific code has been written directly in SageMath for computations that are not covered by pre-existing open source software.

In the particular case of algebraic topology, SageMath includes modules for different classes of complexes (simplicial, Δ, and cubical), and simplicial sets (plus other topological objects of different nature, such as knots, links, and braids for example). From these objects, SageMath can compute invariants such as their homology, both by using custom written code or by leveraging an interface to CHomP [3]. As expected, SageMath only supports this computation over objects of finite type (in general SageMath only supports simplicial sets with a finite number of non-degenerate simplices with one exception: Classifying spaces of finite groups, or more generally, nerves of finite monoids have infinitely many non-degenerate simplices, but finitely many in each dimension). Since Kenzo [7] can handle more general simplicial sets, it is a natural candidate to extend the SageMath functionalities in this aspect.

Indeed, there already exists a way to communicate both programs, but only to construct simplicial sets given a Kenzo output. This communication can be used to work with some particular spaces and one needs to load manually in SageMath some external Kenzo files, i.e., there is no proper connection between both systems (see [12] for further details).

The main contribution of this work is an interface to embed Kenzo and some of its external modules within SageMath, so that both systems can be combined to carry out computations in algebraic topology under a friendly interface by means of Python commands. In addition, an optional package of Kenzo is provided as part of the SageMath distribution. This package can be used as a standalone version of Kenzo, easing its installation from a simple command. We also integrate in the interface and the package an algorithm to compute homotopy groups of simply connected simplicial sets (which are not necessarily 1-reduced). As far as we know, this is the first program that is able to carry out that kind of computation. As an application, the interface allows computing homotopy groups that were not known before.

This way, this work aims to solve the limitations of SageMath on computations in algebraic topology, by integrating most of the computational features of Kenzo, and some of its external modules in SageMath. We also expect to extend the use of Kenzo to a broader community, allowing in this way further new applications in both research and education. To this end, both programs are connected via the ECL library (a library interface to Embeddable Common Lisp). The interface is transparent to the user, i.e., the user does not notice that Kenzo is running within SageMath. This work required the development of new code for both Kenzo and SageMath. Indeed, apart from developing the interface, it has also been necessary to carefully adapt Kenzo code and its extensions to standard Common Lisp in order to make them compatible with ECL.

The germ of this work was presented at conference papers [13,14] (the second one was distinguished with the Best Software Demonstration Award). At that point, the interface was very basic and only homology groups were computed. Now, the work has vastly been improved by adding new functionalities and computations:

- Computation of the join of two simplicial sets, wedges, smash products, homotopy groups, spectral sequences, and finite topological spaces;
- Connection between chain complexes, simplicial sets, and morphisms of chain complexes;
- Improvements and fixes of serious flaws in external packages, which caused erroneous calculations;
- Integration of an algorithm to compute homotopy groups of simply connected (not 1-reduced) simplicial sets from Kenzo to our interface, in order to be able to use it from Sagemath;
- Presentation of some examples of computations combining both systems by means of the interface. That is, the interface allows us to carry out computations that would not be possible using Kenzo and SageMath separately;
- We even show how we computed homotopy groups that were not known before.

The rest of the paper is organized as follows. In Section 2 both the Kenzo system and the effective homology theory are introduced. The SageMath features about algebraic topology are explained in detail in Section 3. The interface that we developed is explained in Section 4, showing computations involving both systems. In Section 5, we present how Kenzo computes homotopy groups, its integration within SageMath and, as an application of our work, the discovering of new homotopy groups. In Section 6, we explain the integration of some Kenzo external modules. Finally, Section 7 presents the conclusions and further work.

2. Effective Homology Theory and the Kenzo System

Kenzo [7] is a symbolic computation system devoted to algebraic topology. It was originally written in 1990 by Sergeraert and Rubio, under the name of EAT (Effective Algebraic Topology) [15]. In 1998 it was rewritten by Sergeraert and Dousson, with the current name of Kenzo. The last available version dates from 2008, but there exists a fork maintained by Heber with compatibility improvements and bugfixes [16].

The topological spaces considered in Kenzo are represented by means of simplicial sets (see [17] for definitions). The main feature of Kenzo is its ability to work with spaces of an infinite nature. To this aim, it uses the Common Lisp programming language and makes an intensive use of functional programming. The program allows in particular the computation of homology and homotopy groups of complicated spaces, for instance, iterated loop spaces of a loop space modified by a cell attachment, components of complex Postnikov towers, etc., which were not known before [9]. Homology and homotopy groups of infinite spaces are computed in Kenzo by means of the effective homology method, introduced in [6] and deeply explained in [9].

The effective homology method is based on the notion of reduction between two chain complexes C_* and D_*, denoted $C_* \Rightarrow D_*$, which is given by three maps $f : C_* \to D_*$, $g : D_* \to C_*$, and $h : C_* \to C_{*+1}$ satisfying some special properties (see [9] for details) such that the homology groups of C_* and D_* are canonically isomorphic. A homotopy equivalence in Kenzo, denoted $C_* \Longleftrightarrow D_*$, is a pair of reductions $C_* \Leftarrow \hat{C}_* \Rightarrow D_*$ (again, a canonical isomorphism $H_*(C) \cong H_*(D)$ is obtained). When an object (for instance, a simplicial set) X is built in Kenzo, an equivalence $C_*(X) \Longleftrightarrow E_*$ is automatically constructed, where $C_*(X)$ is the chain complex associated with X and E_* is a chain complex of finite type (that is, finitely generated in each degree), so that the homology groups of E_* can be determined by means of elementary operations on matrices. In this way, the homology groups of the object X, which can be of an infinite nature, can be obtained thanks to the isomorphism $H_*(X) \cong H_*(C_*(X)) \cong H_*(E)$.

Let us consider as a didactical example the "minimal" simplicial model of the Eilenberg–MacLane space $K(\mathbb{Z},1)$ (see [17]), defined by $K(\mathbb{Z},1)_n = \mathbb{Z}^n$, and constructed in Kenzo as follows.

```
> (setf kz1 (k-z 1))
[K1 Abelian-Simplicial-Group]
```

Let us observe the Common Lisp syntax, where the function `setf` is used to store in the variable `kz1` the result of the function `(k-z 1)`. The Kenzo function `k-z` receives an integer n (in our example, `1`) and returns the Eilenberg–MacLane space $K(\mathbb{Z},n)$.

The abelian simplicial group $K(\mathbb{Z},1)$ has an infinite number of simplices in every dimension ≥ 1, so that the homology groups of `kz1` cannot be elementarily computed. But the effective homology of this object, automatically computed by Kenzo and stored in the slot `efhm`, is an equivalence with the circle:

```
> (efhm kz1)
[K22 Homotopy-Equivalence K1 <= K1 => K16]
> (orgn (k 16))
(CIRCLE)
```

In this way, the homology groups of $K(\mathbb{Z},1)$ are computable through the finite equivalent object K_{16}:

```
> (homology kz1 0 3)
Homology in dimension 0:
Component Z
---done---
Homology in dimension 1:
Component Z
---done---
Homology in dimension 2:
---done---
```

The technique that Kenzo uses to produce automatically the effective homology of an object is based on the following ideas. If an object is of finite type, its effective homology is trivial. For some "primitive" spaces, there are theoretical results that give explicitely an equivalence with a chain complex of finite type (this is the case of the Eilenberg–MacLane space $K(\mathbb{Z},1)$, which is known to have the same homotopy type of the circle $\mathbb{S}^1$). Moreover, for several topological constructions such as twisted cartesian products, tensor products, loop spaces, etc, the effective homology of the constructed object is determined from the effective homology of the initial spaces. In this way, we can construct recursively the effective homology of every space which can be obtained by applying the different algebraic topology constructions available in Kenzo to the primitive cases.

The computation of homotopy groups is dealt in Kenzo by means of the Whitehead tower method [8] (see Section 5 for the description of this algorithm), and it requires the construction of a sequence of fibrations involving spaces of infinite type. As an example, let us consider the Kenzo construction of the Whitehead tower for the computation of the (well-known) result $\pi_6(\mathbb{S}^3) \cong \mathbb{Z}/12\mathbb{Z}$.

```
> (progn
        (setf X (sphere 3))
        (setf k3 (chml-clss X 3))
        (setf t3 (z-whitehead X k3))
        (setf X4 (fibration-total t3))
        (setf k4 (chml-clss X4 4))
        (setf t4 (z2-whitehead X4 k4))
        (setf X5 (fibration-total t4))
        (setf k5 (chml-clss X5 5))
        (setf t5 (z2-whitehead X5 k5))
```

```
        (setf X6 (fibration-total t5))
        (homology X6 6))
Homology in dimension 6:
Component Z/12Z
---done---
```

Despite being the only program for algebraic topology able to carry out computations over infinite structures, the main problem of the Kenzo system is that it requires the use of a Common Lisp IDE and the user should be familiarized with functional programming, as has been seen with the previous examples. Furthermore, Kenzo has been enhanced by several authors with additional modules which are available on the personal websites of the authors, but there is not a stable version including these modules and their installation is not simple. In particular, there exists a module which automatizes the construction of the Whitehead tower of fibrations for the computation of homotopy groups using only one function [18]. In addition, there exists another module computing spectral sequences [19], a useful tool of algebraic topology which in general is not constructive. This module can be applied to determine spectral sequences associated with filtered complexes of infinite type and makes it possible to determine Serre and Eilenberg–Moore spectral sequences associated with fibrations when versions with effective homology are known for the initial spaces. Recently, a new Kenzo module has been developed for computing homotopical invariants of finite topological spaces [20], which can be considered a model for a wide variety of topological spaces.

3. SageMath and Algebraic Topology

SageMath includes modules for constructing and computing properties of topological objects in different flavors: Graphs, knots, braids, manifolds, and algebraic varieties. In particular, some of them overlap with the scope of Kenzo: Chain complexes and simplicial sets. In this section we will oversee a quick survey of the related capabilities.

3.1. Chain Complexes

SageMath supports finite chain complexes graded over a free abelian group. The graded parts of the complex may be free modules over a commutative ring. They can be created by providing the data that defines them in different formats. The simplest ones are either a dictionary with the differential matrices indexed by the elements of the grading group:

```
sage: C = ChainComplex({0: matrix(ZZ, 2, 3, [3, 0, 0, 0, 0, 0])}); C
Chain complex with at most 2 nonzero terms over Integer Ring
sage: ascii_art(C)
            [3 0 0]
            [0 0 0]
 0 <-- C_1 <-------- C_0 <-- 0
```

Or, if the grading group is $\mathbb{Z}$, just a list of matrices:

```
sage: m = matrix(ZZ, 2, 2, [0, -1, 0, 0])
sage: D = ChainComplex([m, m], base_ring=GF(2)); D
Chain complex with at most 3 nonzero terms over Finite Field
of size 2
sage: ascii_art(D)
            [0 1]       [0 1]
            [0 0]       [0 0]
 0 <-- C_2 <------ C_1 <------ C_0 <-- 0
```

Note that, in the last example, since the base ring is the field of two elements, the matrices are converted accordingly.

By default, the grading group is $\mathbb{Z}$ and the degree of the differential is 1. But we can specify a different one:

```
sage: E = ChainComplex({3:m,2:m}, degree_of_differential=-1)
sage: ascii_art(E)
```

```
            [ 0 -1]        [ 0 -1]
            [ 0  0]        [ 0  0]
 0 <-- C_1 <-------- C_2 <-------- C_3 <-- 0
```

Also, new complexes can be created from existing ones:

```
sage: ascii_art(E.dual())
            [ 0  0]        [ 0  0]
            [-1  0]        [-1  0]
 0 <-- C_3 <-------- C_2 <-------- C_1 <-- 0
sage: R.<x,y,z> = QQ[]
sage: C1 = ChainComplex({1:matrix([[x]])},
degree_of_differential=-1)
sage: C2 = ChainComplex({1:matrix([[y]])},
degree_of_differential=-1)
sage: ascii_art(C1.tensor(C2))
                        [ x]
            [y x]       [-y]
 0 <-- C_0 <------ C_1 <----- C_2 <-- 0
```

Once a chain complex has been created, we can compute its homology:

```
sage: C.homology(0)
Z x Z
sage: C.homology(1)
Z x C3
sage: C.homology(2)
0
```

3.2. Simplicial Sets

Simplicial sets with finitely many simplices in each dimension can be created in SageMath either from a library, by applying operations to existing ones, or giving explicitly the non-degenerate simplices and their boundaries. For example, three ways to create the 2-sphere are:

```
sage: S2_1 = simplicial_sets.Sphere(2)
sage: S2_2 = simplicial_sets.Sphere(1).suspension()
sage: from sage.homology.simplicial_set import AbstractSimplex,
SimplicialSet
sage: v = AbstractSimplex(0, name='v')
sage: f = AbstractSimplex(2, name='f')
sage: dv = v.apply_degeneracies(0)
sage: S2_3 = SimplicialSet({f:(dv,dv,dv)})
```

As expected, they have the same Betti numbers:

```
sage: S2_1.betti()
{0: 1, 1: 0, 2: 1}
sage: S2_2.betti()
{0: 1, 1: 0, 2: 1}
sage: S2_3.betti()
{0: 1, 1: 0, 2: 1}
```

The library provides functions to construct some specific cases including classifying spaces, projective spaces (both real and complex), spheres, or the Klein bottle among others:

```
sage: P2 = simplicial_sets.ComplexProjectiveSpace(2); P2
CP^2
```

Moreover, simplicial sets can be created from simplicial complexes or delta complexes, of which SageMath also has a predefined library:

```
sage: from sage.homology.simplicial_set import SimplicialSet
sage: S3 = simplicial_complexes.BarnetteSphere(); S3
Barnette's triangulation of the 3-sphere
sage: KS3 = SimplicialSet(S3); KS3
Simplicial set with 92 non-degenerate simplices
```

Some operations can be used to create new simplicial sets from existing ones, such as the cartesian product, wedge, smash, cone, and suspension. They are all implemented as methods that return a new simplicial set however, the returned object has some extra structure that "remembers" the fact that it was constructed from other pieces, so the different computations of properties and invariants are derived from the ones on the pieces:

```
sage: W = S2_1.wedge(P2); W
Wedge: (S^2 v CP^2)
sage: W.factors()
(S^2, CP^2)
sage: W.all_n_simplices(2)
[rho_0, rho_1, sigma_2, s_1 s_0 *]
sage: S2_1.all_n_simplices(2)
[sigma_2, s_1 s_0 v_0]
sage: P2.all_n_simplices(2)
[rho_0, rho_1, s_1 s_0 v]
```

Several invariants and properties of simplicial sets can be computed by specific methods. Some of the most relevant ones are the underlying chain complex, homology, cohomology (both the cohomology groups and the ring structure), fundamental group, and the algebraic topological model:

```
sage: W.homology(2)
Z x Z
sage: W.homology(3)
0
sage: W.homology(4)
Z
sage: W.cohomology(2)
Z x Z
sage: W.cohomology(3)
0
sage: W.cohomology(4)
Z
sage: ascii_art(W.chain_complex())
                          [ 2  0 -2]       [ 0 -1 -1]
                          [-1  0  1]       [ 1  1  0]
                          [ 0  0  0]       [ 0 -1 -1]
 0 <-- C_0 <-- 0 <-- C_2 <----------- C_3 <----------- C_4 <-- 0
sage: R = W.cohomology_ring(); R
Cohomology ring of Wedge: (S^2 v CP^2) over Rational Field
sage: R.basis()
Finite family {(0, 0): h^{0,0}, (2, 0): h^{2,0}, (2, 1):
h^{2,1}, (4, 0): h^{4,0}}
sage: table([[a*b for b in R.basis()] for a in R.basis()])
  h^{0,0}   h^{2,0}       h^{2,1}   h^{4,0}
  h^{2,0}   1/4*h^{4,0}   0         0
  h^{2,1}   0             0         0
  h^{4,0}   0             0         0
sage: [a.degree() for a in R.basis()]
[0, 2, 2, 4]
sage: simplicial_sets.Torus().fundamental_group()
Finitely presented group < e0, e1 | e0*e1^-1*e0^-1*e1 >
sage: M = W.algebraic_topological_model(); M
(Chain homotopy between:
   Chain complex endomorphism of Chain complex with at most 5
   nonzero terms over Rational Field and Chain complex
   endomorphism of Chain complex with at most 5 nonzero terms
   over Rational Field,
 Chain complex with at most 5 nonzero terms over Rational Field)
```

As we can see, SageMath is quite powerful at computing invariants and properties of simplicial sets. However, it is restricted to simplicial sets with a finite number of simplices in each dimension.

4. The SageMath-Kenzo Interface

In this section we present the optional package to install Kenzo within SageMath and the basics of the interface that connects both systems. The following sections will provide further details on the external packages that we have incorporated and the features.

Our Kenzo package for SageMath is based on the Kenzo version by Heber [16], which is an adapted version of the original Kenzo, but includes improvements to make it compatible with any Common Lisp system. We make it easy to install our Kenzo optional package in SageMath, simply by running:

```
sage -i kenzo
```

This command will download the source code of a fork of Heber's version of Kenzo that we maintain for this specific purpose, compile it inside the ECL interpreter that SageMath includes, and install it. Allegedly, this can be an easy way to install Kenzo even if we just plan to use it as a standalone program. More concretely, we can use Kenzo with an ECL session via the command `sage -ecl`. This way, we could use Kenzo as usual, i.e., by means of Lisp commands.

This Kenzo optional package can also be used within a SageMath session via the `EclObject` and `ecl_eval` functions, which are a wrapper around ECL objects and a function that evaluates a string in Lisp and returns the result, respectively. For instance, the following commands load Kenzo and define the Eilenberg–MacLane space $K(\mathbb{Z}, 3)$ by means of Lisp commands evaluated with `ecl_eval` in a SageMath session.

```
sage: from sage.libs.ecl import ecl_eval
sage: ecl_eval("(require :kenzo)")
sage: ecl_eval("(in-package #:cat)")
sage: ecl_eval("(setf K (k-z 3))")
<ECL: [K25 Abelian-Simplicial-Group]>
```

In the previous example, the execution is carried out in Kenzo via the ECL library, but there is no proper connection to SageMath classes. In addition, this embedding of Kenzo within SageMath does not ease its use: The same Lisp syntax and Kenzo specific commands must be used.

Thus, we decided to develop an interface to communicate both systems. The interface is a Python file that we include in the SageMath distribution. This interface automatically loads Kenzo inside the ECL library, which is itself loaded as a C-library in SageMath. This way, the communication between both systems is done via a C-library interface. It is worth noting that there exist other alternatives to connect both systems, such as pseudo-terminals or temporary files, but a C-library interface has a much lower overhead than them. Moreover, the interface provides functions to create Kenzo objects, and wrappers around them that allow one to create new objects (with the corresponding wrapper) and call functions on them. When it makes sense, the output of those functions is converted to the corresponding SageMath object.

The initial prototype of the interface was available in the SageMath distribution from version 8.7 and included some basic functionalities. In particular, this interface exposed functions to create spheres, Eilenberg–MacLane spaces, and Moore spaces. From these spaces, it allowed the construction of cartesian products, classifying spaces, suspensions, loop spaces, and tensor products. In addition, for each resulting space, the `homology` method can construct the corresponding homology groups.

From SageMath version 9.2, the package and the interface have been enhanced by adding some new functionalities. For instance, we have included some existing external modules of Kenzo that allow computing homotopy groups of 1-reduced simplicial sets (by using the Whitehead tower method) and some spectral sequences. In addition, we have also provided methods for translating objects (such as chain complexes and simplicial sets) both from SageMath to Kenzo and vice versa. This permits one to build simplicial objects

in SageMath, translate them into Kenzo, and compute their homology and homotopy groups. This new version also updates the Kenzo package to fix bugs detected in an external module, to include an algorithm to compute homotopy groups of not 1-reduced simplicial sets, and other features that are explained in detail in the following sections. Furthermore, we also integrate in SageMath, a recent Kenzo module [20] about finite topological spaces. This part of the work (which will be presented in Section 6.3) is not included in SageMath 9.2, but it is currently under review by the Sagemath developers to include in future releases.

Additionally, we provide a website that permits the use of SageMath and Kenzo with the new version of the interface, including the work about finite topological spaces, in an interactive environment. The website permits one to execute commands and explore outputs (https://mybinder.org/v2/gh/jodivaso/examples-sage-kenzo/master?filepath=examples.ipynb accessed on 24 March 2021).

After having reached this link, a standard Jupyter notebook is displayed with the new package enabled and ready for use. This way, the last version of the interface to connect SageMath and Kenzo can be used online with no need to install any software. Such a website also contains all the SageMath code examples presented in this section and the next two ones. The documentation of the new package is available in the official SageMath reference manual [21].

The interface loads Kenzo and disables the standard output. It defines new classes and methods, which are mainly wrappers of the Kenzo ones. By means of Python commands (and calls to the `EclObject` and `ecl_eval` operators) it computes instructions in Kenzo and provides the output as SageMath objects in a transparent way to the user.

Figure 1 presents the class diagram of the interface. The main class is `KenzoObject`, which inherits from `SageObject`, and is a wrapper to a Kenzo object (which is an ECL object).

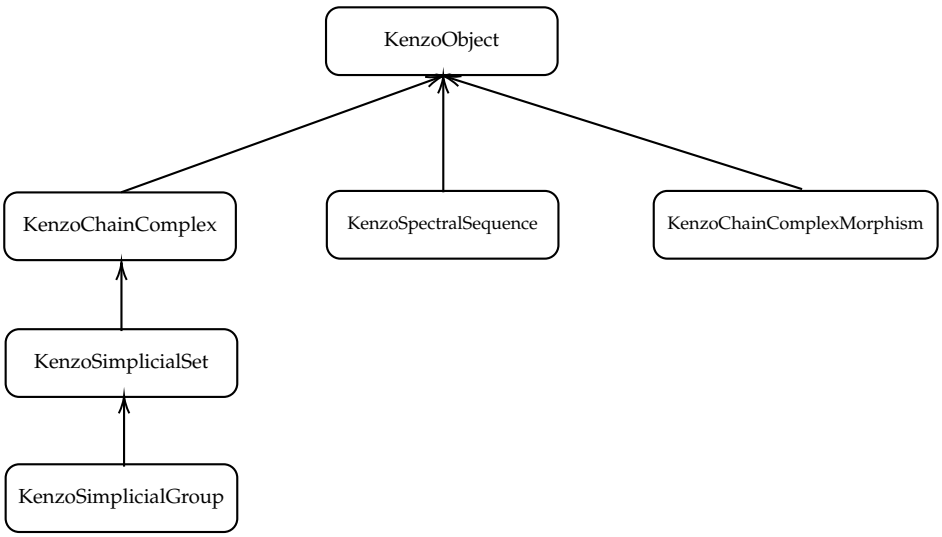

Figure 1. Class diagram of the interface.

The other classes provide specific methods for each type of object (some of them are incorporated to connect the interface to Kenzo external modules). For instance, the `KenzoSimplicialSet` class possesses a method `homotopy_group`, which computes the n-th homotopy group of the simplicial set (via a call to the corresponding function in a Kenzo external package, see Section 5) and prints the result. The method also returns the result as an object of the `AbelianGroup` class, which is part of SageMath. This way, the user could invoke any operation provided by the `AbelianGroup` class over such an object.

It is worth distinguishing between two types of mathematical objects. On the one hand, there are objects (which are necessarily finite) that can be represented in both SageMath and Kenzo. Thus, they can be translated from SageMath to Kenzo and vice versa. For

some of these objects that are available in both Kenzo and SageMath, both systems use the same representation and then the translation between them is easy. This is the case of simplicial sets: An object of the SageMath class `SimplicialSet` can be translated to the wrapper class `KenzoSimplicialSet` defined in the interface via a conversion function called `KFiniteSimplicialSet`. Nevertheless, there are other structures such as chain complexes for which the representations in SageMath and Kenzo are different. As we have explained in Section 3, SageMath can construct a finite chain complex by means of either a dictionary with the differential matrices indexed by the elements of the grading group or, if the grading group is $\mathbb{Z}$, just a list of matrices. However, the Kenzo representation of a finite chain complex consists of a function that provides a list of generators for each degree and the differential maps described also as a function. Thus, both representations share the same information (they represent the same finite chain complex), but in a different format and then a conversion function must be provided. When dealing with this case of different representations, our choice has been to implement the major part of the conversion in Lisp. This permits us to keep the interface as simple as possible and the connection is made via basic data types (integers, strings, lists, etc.). This choice means that the development of the interface has required one to code in both SageMath and Kenzo. In the case of finite chain complexes, we have implemented in Kenzo a function to construct a finite chain complex from a dictionary whose values are the differential matrices, i.e., following the same representation as SageMath uses. Then, the interface just presents the wrapper to such a function.

On the other hand, there are objects that are of an infinite nature, and thus only Kenzo is able to represent them. In this case, the object that represents an infinite space is directly created in SageMath as a wrapper of the corresponding Kenzo object.

For instance, we have defined a function which returns the Eilenberg–MacLane space $K(\pi, n)$ as a Kenzo simplicial group, being π a cyclic group. To do that, we first define in the interface the corresponding ECL objects for $K(\mathbb{Z}, n)$, $K(\mathbb{Z}/2\mathbb{Z}, n)$, and $K(\mathbb{Z}/p\mathbb{Z}, n)$.

```
k_z = EclObject("k-z")
k_z2 = EclObject("k-z2")
k_zp = EclObject("k-zp")
```

Then, those functions are used in the implementation of the wrapper function, which returns an object that belongs to the class `KenzoSimplicialGroup`. That is, the implementation in SageMath internally calls the functions in Kenzo.

```
def EilenbergMacLaneSpace(G, n):
    if G == ZZ:
        kenzospace = k_z(n)
        return KenzoSimplicialGroup(kenzospace)
    elif G == AdditiveAbelianGroup([2]):
        kenzospace = k_z2(n)
        return KenzoSimplicialGroup(kenzospace)
    elif G in CommutativeAdditiveGroups() and G.is_cyclic():
        kenzospace = k_zp(G.cardinality(), n)
        return KenzoSimplicialGroup(kenzospace)
    else:
        raise NotImplementedError("...")
```

Let us note again that we have had to develop code for SageMath (the interface, the wrapper functions), as well as for Kenzo: New functions have been added and some existing ones have been adapted in order to make the output compatible with the interface and SageMath. For example, we have created functions in Kenzo to create finite simplicial sets from the corresponding SageMath objects:

```
sage: from sage.interfaces.kenzo import KFiniteSimplicialSet
sage: CP3 = simplicial_sets.ComplexProjectiveSpace(3); CP3
CP^3
sage: KCP3 = KFiniteSimplicialSet(CP3); KCP3
[K609 Simplicial-Set]
```

Finally, let us present an example of execution in SageMath via Kenzo involving infinite structures. As we have already said, some direct computations of the interface of Kenzo in SageMath include the construction of different elements of algebraic topology which are available in Kenzo such as spheres, Eilenberg–MacLane spaces, cartesian products, tensor products, loop spaces, etc. Although some of them can also be constructed in SageMath as explained in Section 3, Kenzo makes it possible to work with elements of infinite nature so that it is interesting to include these functions in SageMath. For example, we can construct the cartesian product of the Eilenberg–MacLane space $K(\mathbb{Z}/2\mathbb{Z}, 2)$ and the loop space of the sphere $\mathbb{S}^3$ and compute its homology groups as follows (the Python function `range(n)` returns the sequence of numbers $(0, 1, \ldots, n-1)$):

```
sage: from sage.interfaces.kenzo import EilenbergMacLaneSpace
sage: from sage.interfaces.kenzo import Sphere
sage: E = EilenbergMacLaneSpace(AdditiveAbelianGroup([2]), 2)
sage: S3 = Sphere(3)
sage: L = S3.loop_space()
sage: X = E.cartesian_product(L)
sage: [X.homology(i) for i in range(8)]
[Z, 0, Z x C2, 0, Z x C2 x C4, C2, Z x C2 x C2 x C4, C2 x C2]
```

Let us remark that the loop space of $\mathbb{S}^3$ is not of finite type and then the space X is also of an infinite nature. Moreover, the loop space constructor is not available in SageMath. Kenzo is able to compute the homology groups of our space X using the effective homology technique explained in Section 2.

5. Homotopy Groups

The computation of homotopy groups of topological spaces is known to be one of the most challenging problems in the field of algebraic topology. As said in Section 2, making use of the effective homology technique and the Whitehead method [8], Kenzo is able to compute homotopy groups of 1-reduced simplicial sets by means of a tower of fibrations (following the algorithm in [22]). In the Kenzo external module [18], a function is provided which automatizes this construction and computes the homotopy group of a 1-reduced simplicial set for a given dimension. However, this external module is not able to work with spaces which are not 1-reduced. Moreover, we have found several bugs in this module in different situations. Taking into account that SageMath users could be interested in computing homotopy groups of spaces which are not 1-reduced, we have decided not only to include the external module [18] in our interface, but also to correct it and enhance it so that we can compute homotopy groups of simply-connected spaces which are not necessarily 1-reduced.

The Whitehead tower method [8] for computing homotopy groups is based on the following ideas. Given $n \geq 2$, let X be an $(n-1)$-connected simplicial set, that is, such that $H_0(X) = \mathbb{Z}$ and $H_i(X) = 0$ for $1 \leq i \leq n-1$. Let us suppose $H_n(X) \neq 0$, then the Hurewicz theorem [23] claims that $\pi_i(X) = 0$ for all $1 \leq i \leq n-1$ and $\pi_n(X) \cong H_n(X)$. The Whitehead tower method consists in producing a new space X_{n+1} and a fibration with fiber G_{n+1} being an Eilenberg–MacLane space:

$$G_{n+1} \equiv K(H_n(X), n-1) \longrightarrow X_{n+1} \longrightarrow X$$

such that $\pi_i(X_{n+1}) = 0$ for $1 \leq i \leq n$ and $\pi_i(X_{n+1}) \cong \pi_i(X)$ for $i \geq n+1$. In particular, $\pi_{n+1}(X) \cong \pi_{n+1}(X_{n+1}) \cong H_{n+1}(X_{n+1})$. The process is iterated over X_{n+1}, producing a tower of fibrations:

$$\begin{array}{ccc} & & \vdots \\ & & \downarrow \\ G_{n+3} \equiv K(H_{n+2}(X_{n+2}), n+1) & \longrightarrow & X_{n+3} \\ & & \downarrow \\ G_{n+2} \equiv K(H_{n+1}(X_{n+1}), n) & \longrightarrow & X_{n+2} \\ & & \downarrow \\ G_{n+1} \equiv K(H_n(X), n-1) & \longrightarrow & X_{n+1} \\ & & \downarrow \\ & & X \end{array}$$

such that each X_k in the tower satisfies $\pi_k(X_k) \cong H_k(X_k) \cong \pi_k(X)$, for $k \geq n+1$. In this way, if we are able to compute all homology groups $H_k(X_k)$, the homotopy groups $\pi_k(X)$ can be determined. However, let us emphasize that if some $H_k(X_k)$ appearing in the construction are not finite (for instance, $\mathbb{Z}$), then the Eilenberg–MacLane space $K(H_k(X_k), k-1)$ and then the corresponding total space X_{k+1} are not of finite type, so that its homology groups cannot be directly determined and the construction of the tower cannot always be completed. This problem is solved using the effective homology theory as follows.

The Whitehead fibrations are implemented in the Kenzo system for simplicial sets X which are 1-reduced, that is, which have only one vertex (0-simplex) and no non-degenerate 1-simplex. More concretely, let X be a connected $(n-1)$-reduced simplicial set, that is, it has only one 0-simplex and it does not contain any non-degenerate i-simplex for $1 \leq i \leq n-1$, which implies $H_0(X) = \mathbb{Z}$ and $H_i(X) = 0$ for $1 \leq i \leq n-1$. Let us suppose that $H_n(X) \cong \mathbb{Z}$ or $H_n(X) \cong \mathbb{Z}/m\mathbb{Z}$ for some $m \geq 2$. In both cases, (different) Kenzo functions are available which input the degree n and the simplicial set X and construct the Whitehead fibration:

$$G_{n+1} \equiv K(H_n(X), n-1) \longrightarrow X_{n+1} \longrightarrow X.$$

The process can be iterated to obtain the whole tower of fibrations. However, the only Eilenberg–MacLane spaces $K(\pi, n)$, which are available in Kenzo and its external modules are those with $\pi = \mathbb{Z}$ or $\pi = \mathbb{Z}/m\mathbb{Z}$ for some $m \geq 2$. When some homology group $H_k(X_k)$ is obtained such that it is a sum of several factors $\mathbb{Z}$ or $\mathbb{Z}/m\mathbb{Z}$, $H_k(X_k) \equiv H_k^1(X_k) \oplus \cdots \oplus H_k^r(X_k)$, then it is known that:

$$K(H_k(X_k), k-1) \cong K(H_K^1(X_k), k-1) \times \cdots \times K(H_K^r(X_k), k-1)$$

and the fibration:

$$G_{k+1} \equiv K(H_k(X_k), k-1) \longrightarrow X_{k+1} \longrightarrow X_k$$

in the tower can be simulated by several fibrations as follows:

$$
\begin{array}{c}
\vdots \\
\downarrow \\
G_{k+2} \equiv K(H_{k+1}(X_{k+1}), k) \longrightarrow X_{k+2} \\
\downarrow \\
G_{k+1} \equiv G^r_{k+1} \equiv K(H^r_k(X_k), k-1) \longrightarrow X^r_{k+1} \equiv X_{k+1} \\
\downarrow \\
\cdots \\
\downarrow \\
G^2_{k+1} \equiv K(H^2_k(X_k), k-1) \longrightarrow X^2_{k+1} \\
\downarrow \\
G^1_{k+1} \equiv K(H^1_k(X_k), k-1) \longrightarrow X^1_{k+1} \\
\downarrow \\
G_k \equiv K(H_{k-1}(X_{k-1}), k-2) \longrightarrow X_k \\
\downarrow \\
\cdots
\end{array}
\tag{1}
$$

Again, let us observe that if some $H_k(X_k)$ or $H^j_k(X_k)$ appearing in the construction are not finite then the space $K(H_k(X_k), k-1)$ (or $K(H^j_k(X_k), k-1)$) and the total space X_{k+1} are not of finite type. However, if the simplicial set X is finite or has effective homology, then Kenzo computes in an automatic way the effective homology of the spaces X_k or X^j_k and then their homology groups can be determined. If we want to compute the s-th homotopy group $\pi_s(X)$, it is sufficient to construct the tower up to dimension s (the space X_s satisfies $\pi_s(X) \cong \pi_s(X_s) \cong H_s(X_s)$).

The external module [18] automatizes the construction of all the fibrations in the tower in a recursive way and provides a function for computing the s-th homotopy group of a (1-reduced) space. However, we have found three bugs in that external module, which we have solved as follows:

- When some homology group $H_k(X_k)$ in the tower is obtained such that it is a sum of several factors $\mathbb{Z}$ or $\mathbb{Z}/m\mathbb{Z}$, $H_k(X_k) \equiv H^1_k(X_k) \oplus \cdots \oplus H^r_k(X_k)$, the intermediate spaces $K(H^j_k(X_k), k-1)$ in diagram (1) are not $(k-1)$-reduced. To solve this problem, it is necessary to introduce a new step in the algorithm to produce a reduction from $K(H^j_k(X_k), k-1)$ to a $(k-1)$-reduced space. This step was already available in Kenzo but was not used in the external module [18], so that incorrect results were obtained. We have solved this problem by using the corresponding Kenzo function producing the necessary reduction;
- When some homology group $H_k(X_k)$ is equal to zero, the external module [18] stops the construction of the tower and claims that $\pi_i(X) = 0$ for every $i \geq k$. This assumption is incorrect since there exist spaces with $\pi_k(X) = 0$ but $\pi_i(X) \neq 0$ for some $i > k$. We have solved this bug by continuing the construction of the tower from the following step such that $H_i(X_k) \neq 0$;
- Each fibration $K(H_k(X_k), k-1) \longrightarrow X_{k+1} \longrightarrow X_k$ in the tower is defined by means of a simplicial morphism of degree -1, called the twisting operator [17], $\tau_k : X_K \to K(H_k(X_k), k-1)$. This simplicial morphism is described by a well-known and existing isomorphism between $H_k(X_k)$ and $K(H_k(X_k), k-1)_{k-1}$ (the set of the $(k-1)$-simplices of $K(H_k(X_k), k-1)$). In order to construct this twisting operator, Kenzo uses the Smith

Normal Form (SNF) [24] of the differential matrix of degree $k+1$ of the simplicial set X_k, denoted D^k_{k+1}. The SNF algorithm of a matrix A consists in computing a diagonal matrix D such that each diagonal element $d_{i,i}$ divides $d_{i+1,i+1}$ and two invertible matrices (corresponding to the changes of bases), P and Q, such that $PAQ = D$. In particular, the SNF algorithm of D^k_{k+1} produces a diagonal matrix whose entries describe the (factors of the) group $H_k(X_k)$. If $H_k(X_k)$ is a sum of several factors, the intermediate fibrations $K(H^j_k(X_k), k-1) \longrightarrow X^j_{k+1} \longrightarrow X^{j-1}_{k+1}$ must be constructed considering the diagonal elements of the SNF (which is unique). However, the external module [18] uses a different decomposition of $H_k(X_k)$ (by means of a diagonal matrix which also describes the homology group but does not always satisfy the SNF property), so that in some situations an error is found. We have solved this bug by transforming the homology group factors used in the module into SNF, constructing the intermediate fibrations following the correct order.

After solving the previous bugs and enhancing the external module [18] for computing homotopy groups, we have included it in our interface for SageMath. To this aim, the first step has been to make the Lisp code valid for ECL. Later, we have written SageMath functions and wrappers, so that one can compute homotopy groups of (simply connected) simplicial sets in SageMath.

Then, we have also considered the situation where the initial simplicial set X is not 1-reduced but only simply connected (that is, such that $H_1(X) = 0$ but there exist non-degenerate simplices of dimension 1). In that situation, the twisting operator $\tau_k : X_K \to K(H_k(X_k), k-1)$ corresponding to the existing isomorphism between $H_k(X_k)$ and $K(H_k(X_k), k-1)_{k-1}$ cannot be as easily determined as before. However, Kenzo's algorithm can be modified by considering the two differential matrices of X_k of degrees k and $k+1$, denoted D^k_k and D^k_{k+1} respectively, and following the same steps used for computing homology groups. This new algorithm is described in the preprint [25] and can be of great interest for computing homotopy groups of spaces which are simply connected, but not 1-reduced. This is the case, as we will see in the following subsection, of simplicial complexes which can be constructed in SageMath coming from different areas. We also integrated it in our interface for SageMath.

In the following lines, we show an example of the computation of homotopy groups of the wedge of the sphere $\mathbb{S}^3$ and the suspension of the torus (the Python function `range(n,m)` returns the sequence of numbers $(n, n+1, \ldots, m-1)$).

```
sage: from sage.interfaces.kenzo import KFiniteSimplicialSet
sage: S3 = simplicial_sets.Sphere(3)
sage: T = simplicial_sets.Torus()
sage: sT = T.suspension()
sage: S3vsT = S3.wedge(sT)
sage: K = KFiniteSimplicialSet(S3vsT)
sage: [K.homotopy_group(i) for i in range(2,5)]
[Multiplicative Abelian group isomorphic to Z x Z,
Multiplicative Abelian group isomorphic to Z x Z x Z x Z x Z,
Multiplicative Abelian group isomorphic to Z x Z x Z x Z x Z
x Z x C2 x C2 x C2 x C2 x C2]
```

Let us note that both systems are collaborating: The sphere, the torus, the suspension, and wedge product are computed by means of functions that are available in SageMath, whereas the computation of the homotopy groups of the constructed simplicial object is done in Kenzo, since the simplicial set library provided by SageMath cannot deal with that kind of computation. Let us also emphasize that the wedge construction produces a simplicial set wich is not 1-reduced.

Discovering New Homotopy Groups

In the following example, we will show how the collaboration of SageMath and Kenzo allowed us to compute some homotopy groups that were previously unknown.

Given a set V of n vertices, the set of possible pairs P has $\binom{n}{2}$ elements. A subset $S \subseteq P$ determines a graph on the vertices (where the elements of S are the edges of the graph). For an integer i, a graph is said to be i-connected if it cannot become disconnected by removing less than i vertices. It is clear that if $S' \subseteq S$ and S is not i-connected, then S' is also not i-connected. So, the set of all not i-connected graphs on n vertices forms a simplicial complex, denoted as Δ_n^i.

In [26], these simplicial complexes have been studied in relation to Knot Vasiliev invariants and some theoretical results about their homotopy type have been stated:

- It is proved that Δ_n^1 has the homotopy type of a wedge of $(n-1)!$ spheres of dimension $n-3$;
- It is proved that, if $n \geq 3$, Δ_n^2 has the homotopy type of a wedge of $(n-2)!$ spheres of dimension $2n-5$;
- It is conjectured that Δ_n^3 has the homotopy type of a wedge of $(n-3)(n-2)!/2$ spheres of dimension $2n-4$.

SageMath can explicitely construct these complexes, and compute some of their invariants, such as the homology:

```
sage: NI = simplicial_complexes.NotIConnectedGraphs(5,3); NI
Simplicial complex of not 3-connected graphs on 5 vertices
sage: [NI.homology(i) for i in range(1,8)]
[0, 0, 0, 0, 0, Z^6, 0]
```

Now, thanks to the interface with Kenzo, we can also compute its homotopy groups, but we must have some care to increase the size of the memory heap that will be used:

```
sage: import sage.libs.ecl
sage: from sage.homology.simplicial_set import SimplicialSet
sage: from sage.interfaces.kenzo import KFiniteSimplicialSet
sage: sage.libs.ecl.ecl_eval("(ext:set-limit 'ext:heap-size 0)")
sage: SNI = SimplicialSet(NI)
sage: KNI = KFiniteSimplicialSet(SNI)
sage: pi6 = KNI.homotopy_group(6); pi6
Multiplicative Abelian group isomorphic to Z x Z x Z x Z x Z x Z
```

This is the expected result by Hurewicz's theorem [23], but we can go further:

```
sage: pi7 = KNI.homotopy_group(7); pi7
Multiplicative Abelian group isomorphic to C2 x C2 x C2 x C2 x C2 x C2
```

which, to our knowledge, was not known up to now. This computation took more than 14 h in a computing node with Intel Xeon E5-2650 v4, 2.20GHz processors, and 512 GB of RAM.

Other homotopy groups π_m of spaces Δ_n^i obtained thanks to our Kenzo interface in SageMath and the integrated algorithm for computing (non 1-reduced) homotopy groups are shown in Table 1. Let us remark that the groups $\pi_m(\Delta_4^2)$ could be theoretically determined thanks to the results of [26], but the group $\pi_8(\Delta_6^3)$ was also unknown. One can also observe that all our results support the conjecture in [26].

Table 1. Homotopy groups of the complex of not i-connected graphs on n vertices.

K	m	$\pi_m(K)$
Δ_4^2	4	C_2^2
Δ_4^2	5	$\mathbb{Z} \times C_2^2$
Δ_4^2	6	$C_2 \times C_{12}^2$
Δ_6^3	8	$\mathbb{Z}^{36}$

6. Integration of Other External Modules

This section presents how we have integrated in our development other external Kenzo modules, concretely the following ones:

1. Joins and wedges [27], developed by Heras, 2010;
2. Spectral sequences [19], developed by the fourth author of this paper, 2006;
3. Finite topological spaces [20], developed by the first author of this paper, 2020.

We also explain new Kenzo functions (and their integration in SageMath) for working with smash products. Some examples of execution are also shown.

6.1. Joins, Wedges, and Smash Products

The join and the wedge of two simplicial sets are two useful constructors in algebraic topology which are implemented in an external module of Kenzo as particular cases of pushout (see [27]). The wedge construction is also implemented in SageMath for finite simplicial sets (as seen in the first example in Section 5), but we have considered including the wedge Kenzo construction so that it can be applied to spaces of an infinite nature. In those cases, the effective homology method can be used to determine the homology groups and other topological invariants. The join construction is not available in SageMath, so that it is also convenient to include it in our interface.

Since Kenzo functions for wedge and join are included in an external module (and not in the standard version), again, the first step to make them available in SageMath has been to modify the Lisp code and make it valid for ECL. Then, we have defined SageMath functions for computing wedges and joins as follows.

First of all, we consider the wedge of the torus with the loop space of the sphere $\mathbb{S}^2$.

```
sage: from sage.interfaces.kenzo import Sphere
sage: from sage.interfaces.kenzo import KFiniteSimplicialSet
sage: T1 = simplicial_sets.Torus()
sage: T = KFiniteSimplicialSet(T1)
sage: S2 = Sphere(2)
sage: lS2 = S2.loop_space()
sage: TvlS2 = T.wedge(lS2)
sage: [TvlS2.homology(i) for i in range(1,5)]
[Z x Z x Z, Z x Z, Z, Z]
```

Let us note that once again both systems are collaborating for the task: The torus is computed by means of a function which is available in SageMath (and not in Kenzo), whereas the sphere, the loop space, the wedge, and the computation of the homology groups of the constructed simplicial object are carried out via Kenzo. As in the last example of Section 4, the final object is of an infinite nature and then SageMath is not able to determine its homology groups; the effective homology method is used in Kenzo.

Similarly, we can construct the joins of simplicial sets of an infinite nature and determine their homology and, when the result space is simply connected, homotopy groups. For instance, we consider two Eilenberg–MacLane spaces.

```
sage: from sage.interfaces.kenzo import EilenbergMacLaneSpace
sage: E1 = EilenbergMacLaneSpace(ZZ, 2)
sage: E2 = EilenbergMacLaneSpace(ZZ, 3)
sage: E = E1.join(E2)
sage: [E.homology(i) for i in range(11)]
[Z, 0, 0, 0, 0, 0, Z, 0, Z x C2, 0, Z x C2 x C3]
sage: [E.homotopy_group(i) for i in range(6, 9)]
[Multiplicative Abelian group isomorphic to Z,
 Trivial Abelian group,
 Multiplicative Abelian group isomorphic to Z]
```

The smash product of two simplicial sets is another topological construction which can be considered as a particular example of pushout. This construction was not included in the external module [27], but we have used the general definition of pushout to develop new Kenzo functions producing the construction of the smash product of two simplicial sets and its effective homology, which makes it possible to determine its homology and homotopy groups. Considering again the two Eilenberg–MacLane spaces of the previous example, we obtain the following results.

```
sage: E3 = E1.smash_product(E2)
sage: [E3.homology(i) for i in range(11)]
[Z, 0, 0, 0, 0, Z, 0, Z x C2, 0, Z x C2 x C3, C2]
sage: [E3.homotopy_group(i) for i in range(5, 9)]
[Multiplicative Abelian group isomorphic to Z,
 Trivial Abelian group,
 Multiplicative Abelian group isomorphic to Z,
 Multiplicative Abelian group isomorphic to C2]
```

6.2. Spectral Sequences

As we have already said in Section 2, there exists a Kenzo module [19] for computing spectral sequences, a tool of algebraic topology given by a family of "pages" of groups $\{E^r_{p,q}\}_{r\geq 1, p\in\mathbb{Z}, q\in\mathbb{Z}}$ and differential maps $d^r_{p,q} : E^r_{p,q} \to E^r_{p-r,p+r-1}$ ([28]). This module is able to determine spectral sequences of filtered complexes of infinite type and, in particular cases, it can compute Serre and Eilenberg–Moore spectral sequences of fibrations with objects of infinite type when versions with effective homology are known for the initial spaces. It can also compute the spectral sequence associated with a bicomplex (see [28] for the definition of these spectral sequences). The module determines not only the groups, but also the differential maps of every page of the spectral sequence.

We have integrated in SageMath the computation of the following spectral sequences:

- Spectral sequence of a bicomplex given by a list of morphisms of chain complexes;
- Serre spectral sequence associated with a cartesian product;
- Serre spectral sequence associated with the first fibration in the Whitehead tower of a simplicial set;
- Eilenberg–Moore spectral sequence associated with the loop space of a simplicial set.

To this aim, again, we have adapted the code of [19] so that it is compatible with ECL and have developed new functions in both Lisp and SageMath. In the four cases, the result is an object of a new class `KenzoSpectralSequence`, for which we can obtain groups, differential matrices, and tables showing the groups of a given level via suitable methods. Moreover, we have defined a new SageMath class called `KenzoChainComplexMorphism` (see Figure 1), which is used to construct the spectral sequence associated with a bicomplex as we will explain later, and a method to translate a chain complex morphism from SageMath to Kenzo.

First of all, we can compute spectral sequences associated with bicomplexes, a particular case of filtered complex from which many useful spectral sequences are defined. A bicomplex (or double complex) $C_{*,*}$ is a bigraded free $\mathbb{Z}$-module $C_{*,*} = \{C_{p,q}\}_{p,q\in\mathbb{Z}}$ provided with morphisms $d'_{p,q} : C_{p,q} \to C_{p-1,q}$ and $d''_{p,q} : C_{p,q} \to C_{p,q-1}$ satisfying $d'_{p-1,q} \circ d'_{p,q} = 0, d''_{p,q-1} \circ d''_{p,q} = 0$, and $d'_{p,q-1} \circ d''_{p,q} + d''_{p-1,q} \circ d'_{p,q} = 0$. We consider as an example the bicomplex of Figure 2 where the differential maps $d'_{1,1}$, $d'_{2,0}$, and $d''_{1,1}$ are given by the matrices $D'_{1,1} = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$, $D'_{2,0} = \begin{pmatrix} 2 & 0 \end{pmatrix}$, and $D''_{1,1} = \begin{pmatrix} 1 & 0 \end{pmatrix}$.

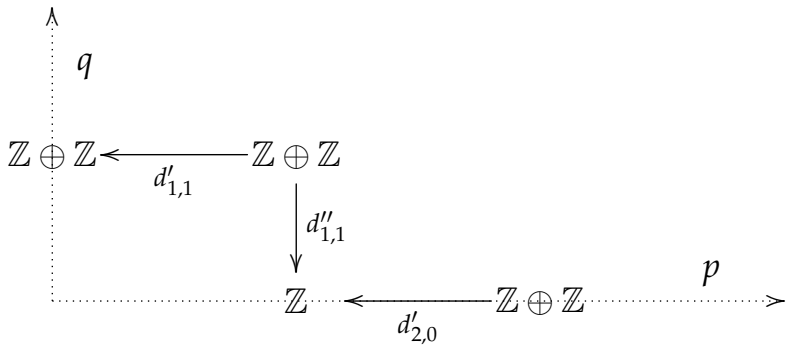

Figure 2. Example of bicomplex.

The associated spectral sequence is represented in Figure 3, with differential maps $d^1_{1,1} : E^1_{1,1} = \mathbb{Z} \to E^1_{0,1} = \mathbb{Z} \oplus \mathbb{Z}$ given by the matrix $\begin{pmatrix} 0 \\ 2 \end{pmatrix}$ and $d^2_{2,0} : E^2_{2,0} = \mathbb{Z} \oplus \mathbb{Z} \to E^2_{0,1} = \mathbb{Z}/2\mathbb{Z} \oplus \mathbb{Z}$ defined by $\begin{pmatrix} 0 & 0 \\ -4 & 0 \end{pmatrix}$.

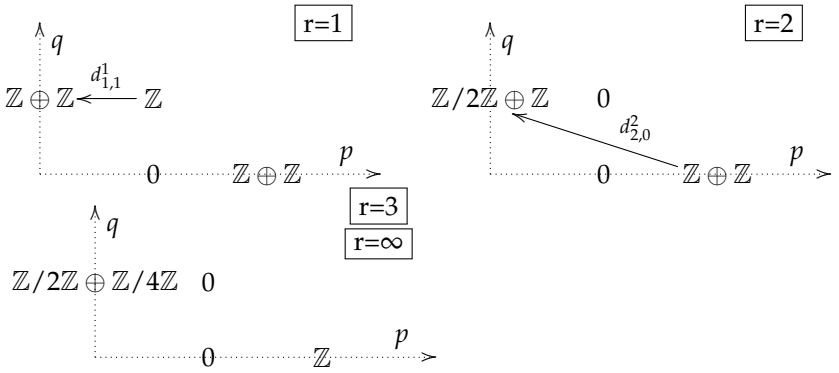

Figure 3. Spectral sequence of the bicomplex.

For the moment, we have made it possible to construct the spectral sequence of a bicomplex in SageMath by means of a list of chain complex morphisms, where each chain complex is a column of the bicomplex (with vertical differential maps) and horizontal differentials are given by the morphisms (with the corresponding signs for vertical maps such that the property of bicomplex is satisfied). Considering the example of Figure 2, we construct the chain complexes corresponding to the three columns and then two chain complex morphisms as follows:

```
sage: C1 = ChainComplex({1: matrix(ZZ, 0, 2, [])},
degree_of_differential=-1)
sage: C2 = ChainComplex({1: matrix(ZZ, 1, 2, [1, 0])},
degree_of_differential=-1)
sage: C3 = ChainComplex({0: matrix(ZZ, 0,2 , [])},
degree_of_differential=-1)
sage: M1 = Hom(C2,C1)({1: matrix(ZZ, 2, 2, [2, 0, 0, 2])})
sage: M2 = Hom(C3,C2)({0: matrix(ZZ, 1, 2, [2, 0])})
```

Let us emphasize that all of them are objects built using SageMath functions. Then, we use our new function `BicomplexSpectralSequence`, which produces an object of the class `KenzoSpectralSequence`, and we determine some of the groups and differential maps. In this class, the method `group` receives three parameters r, p, and q and computes the group $E^r_{p,q}$ of the spectral sequence, in our examplex, $E^2_{0,2} = \mathbb{Z}/2\mathbb{Z} \oplus \mathbb{Z}$. The method `table` inputs five integers: The page r and four indexes $i1$, $i2$, $j1$, and $j2$ corresponding to the first column, last column, first row, and last row to print; in this case we show some groups of the page E^3. The method `matrix` computes the differential map $d^r_{p,q}$ as a matrix and as said before, $d^2_{2,0}$ is given by the matrix $\begin{pmatrix} 0 & 0 \\ -4 & 0 \end{pmatrix}$.

```
sage: from sage.interfaces.kenzo import BicomplexSpectralSequence
sage: l = [M1, M2]
sage: E = BicomplexSpectralSequence(l)
sage: E.group(2,0,1)
Additive abelian group isomorphic to Z/2 + Z
sage: E.table(3,0,2,0,2)
  0           0   0
  Z/2 + Z/4   0   0
  0           0   Z
```

```
sage: E.matrix(2,2,0)
[0  0]
[-4 0]
```

It would also be possible to define a class `Bicomplex` (which does not exist either in SageMath or Kenzo), providing methods for obtaining the bigraded list of generators, vertical and horizontal differentials, and the associated total chain complex.

Another type of spectral sequence that we have integrated in SageMath is the Serre spectral sequence [29], which provides information on the homology groups of the total space of a fibration from the homologies of the base and the fiber. A particular case is obtained for trivial fibrations, which correspond to cartesian products of two simplicial sets. Let us consider for example the cartesian product of two spheres, $\mathbb{S}^2 \times \mathbb{S}^3$. As it is well known, the page E^2 of the Serre spectral sequence is given by $E^2_{p,q} = H_p(\mathbb{S}^3, H_q(\mathbb{S}^2))$ and all differential maps are null (since the fibration is trivial). In the following lines, we compute for instance $E^2_{3,0} = \mathbb{Z}$, we show some of the groups of the page E^2, and we determine the differential map $d^3_{3,0}$ which as expected is the null morphism.

```
sage: from sage.interfaces.kenzo import Sphere
sage: S2 = Sphere(2)
sage: S3 = Sphere(3)
sage: P = S3.cartesian_product(S2)
sage: E = P.serre_spectral_sequence()
sage: E.group(2,3,0)
Additive abelian group isomorphic to Z
sage: E.table(2, 0, 3, 0, 2)
Z       0       0       Z
0       0       0       0
Z       0       0       Z
sage: E.matrix(3,3,0)
[0]
```

Let us consider now the Serre spectral sequence of the Whitehead tower of the sphere $\mathbb{S}^3$, which corresponds to a (non-trivial) fibration $K(\mathbb{Z},2) \to X \to \mathbb{S}^3$. In this case, it is known that the E^2 page is given by $E^2_{p,q} = H_p(\mathbb{S}^3, H_q(K(\mathbb{Z},2)))$ and some non null differential maps can happen. In the following lines, we show the computation of the page E^2 and some differentials. We can observe that $d^3_{3,0} : E^3_{3,0} \to E^3_{0,2}$ is an isomorphism and $d^3_{3,2} : E^3_{3,2} \to E^3_{0,4}$ is the multiplication by 2.

```
sage: from sage.interfaces.kenzo import Sphere
sage: S3 = Sphere(3)
sage: E = S3.sw_spectral_sequence()
sage: E.table(2, 0, 3, 0, 4)
Z       0       0       Z
0       0       0       0
Z       0       0       Z
0       0       0       0
Z       0       0       Z
sage: E.matrix(3,3,0)
[1]
sage: E.matrix(3,3,2)
[2]
```

Another classical example of spectral sequence is the Eilenberg–Moore spectral sequence [30], which is also associated with fibrations, and in particular the one related with the loop space of a simplicial set which is implemented in the Kenzo module [19] and has been also integrated in SageMath. This is a second quadrant spectral sequence, which means that the groups $E^r_{p,q}$ which are different from zero satisfy $p \leq 0$ and $q \geq 0$. Let us consider as an example the computation of the Eilenberg–Moore spectral sequence of the sphere $\mathbb{S}^3$, where we can find for instance a differential map $d^1_{-2,8} : E^1_{-2,8} = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z} \to E^1_{-3,8} = \mathbb{Z} \oplus \mathbb{Z} \oplus \mathbb{Z}$.

```
sage: from sage.interfaces.kenzo import Sphere
sage: S3 = Sphere(3)
sage: L = S3.loop_space()
sage: EMS = L.em_spectral_sequence()
sage: EMS.table(1, -5, -2, 5, 8)
0   Z   Z + Z + Z   Z + Z + Z
0   0   0           0
0   0   Z           Z + Z
0   0   0           0
sage: EMS.matrix(1, -2 ,8)
[ 3 -2  0]
[ 3  0 -3]
[ 0  2 -3]
```

6.3. Finite Topological Spaces

A finite topological space (or simply a finite space) is a topological space with finitely many points. Finite spaces and finite preordered sets are basically the same objects considered from different perspectives [31] (moreover, finite T_0-spaces correspond to finite partially ordered sets or posets). This result allows us to study finite spaces by combining algebraic topology with the combinatorics arising from their intrinsic preorder structures.

The study of homotopical invariants can be restricted to the class of finite T_0-spaces, since any finite topological space is known to be homotopy equivalent to a finite T_0-space [32]. Given a finite T_0-space, the incidence matrix corresponding to the order relation of its associated poset and the adjacency matrix of the Hasse diagram are called the topogenous matrix and Stong matrix of the space, respectively.

An external module of the Kenzo system has been recently developed in order to represent and compute some topological invariants of finite topological spaces [33]. Since many of the algorithms are based on manipulations of the topogenous and Stong matrices, finite spaces are represented in Kenzo as instances of the class `FINITE-SPACE` by using the slots `top` (the topogenous matrix), `stong` (the Stong matrix), and `heights` (a list of the elements of the space separated by heights). An instance of `FINITE-SPACE` can be defined by just declaring either the `top` slot or the `stong` slot, but it is preferable to keep both in order to facilitate the computations. For example, we can construct a finite T_0-space by specifying the edges of the Hasse diagram of its associated poset and constructing the respective Stong matrix:

```
> (setf edges (make-array 19 :initial-contents '(() () () ()
(1 2) (1 2) (2 3) (2 4) (2 3) (2 4) (3 4) (3 4) (5 6) (5 6 7)
(5 6 7 8) (7 8 11) (8 9 12) (9 10 11) (7 10 12))))
> (setf finite-space (build-finite-space
                       :stong (edges-to-stong-mtrx edges)
                       :orgn '(example)))
[K1 Finite-Space]
```

In a particular case, when a finite T_0-space is h-regular, some algorithms to compute its homology have been developed in [33], where the use of discrete vector fields has been also considered in order to deal with smaller chain complexes when computing homology groups. Regarding `finite-space` (the space we have constructed above), we can prove that such a space is h-regular, and then we can compute its homology in Kenzo:

```
> (h-regular-homology finite-space 0)
Component Z
> (h-regular-homology finite-space 1)
Component Z/2Z
> (h-regular-homology finite-space 2)
Component Z
Component Z
```

Despite the capabilities of Kenzo to compute topological invariants, it lacks a graphical interface, as mentioned before. Moreover, finite T_0-spaces can be represented by their associated posets and since the class `FinitePoset` [34] in SageMath is devoted to posets, we have decided to integrate in SageMath the external Kenzo module dealing with finite spaces. By means of this integration, we can use the methods and functions implemented in the class `FinitePoset`, which in particular allows us to visualize the Hasse diagrams of the posets associated to finite T_0-spaces by using graphic objects in SageMath and we enhanced SageMath with a new mathematical object, expanding the scope of this system.

In order to deal with finite spaces in SageMath, we have created a module implementing finite topological spaces and related concepts. The class `FiniteTopologicalSpace` has been created by taking as input the parameters `elements` (the list of elements of the finite space), `minimal_basis` (a dictionary where the keys are the points `n` in `elements` and `minimal_basis[n]` is the set of points in the minimal open set containing `n`), and `topogenous` (the topogenous matrix of the finite space induced by the order given in `elements`). Bearing in mind the importance of finite T_0-spaces in the computation of invariants, we have created the subclass `FiniteTopologicalSpace_T0`, inheriting all the attributes above mentioned and, additionally, the parameter `poset`, a `FinitePoset` representing the poset corresponding to the finite T_0-space.

The function `FiniteSpace` has been implemented to create a finite space in SageMath by using different types of data. For example, by specifying a dictionary representing a minimal basis, we can construct a finite space:

```
sage: from sage.homology.finite_topological_spaces
      import FiniteSpace
sage: minimal_basis = {5: {4, 5}, 4: {4}, 2: {2}, 6: {2, 4, 6},
                       1: {1, 2, 4}, 3: {1, 2, 4, 3}}
sage: M = FiniteSpace(minimal_basis)
```

We have implemented functions to compute some topological properties. For instance, the module provides functions to determine if a point is interior, exterior, boundary, limit, closure, or isolated with respect to a given subspace. Moreover, for a given subspace, the corresponding interior, exterior, boundary, derived, closure, or isolated set can be computed. Regarding point reductions techniques, identification of beat points and weak points [35] have been also implemented in order to compute, for example, cores (the smallest strong deformation retracts [32]) of a finite T_0-space:

```
sage: M.core()
Finite T0 topological space of 4 points with minimal basis
{2: {2}, 3: {2, 4, 3}, 4: {4}, 6: {2, 4, 6}}
```

The point reduction techniques referred to above have been implemented by using the corresponding functionalities in Kenzo as well as the computation of discrete vector fields and homology of h-regular finite T_0-spaces:

```
sage: covers = [[9, 13], [7, 13], [4, 13], [8, 12], [7, 12], [5, 12],
                [9, 11], [6, 11], [5, 11], [8, 10], [6, 10], [4, 10],
                [3, 9], [2, 9], [3, 8], [2, 8], [3, 7], [1, 7],
                [3, 6], [1, 6], [2, 5], [1, 5], [2, 4], [1, 4]]
sage: P = Poset((list(range(1,14)), covers), cover_relations = True)
sage: X = FiniteSpace(P)
sage: dvf = X.discrete_vector_field()
sage: X.hregular_homology(dvfield = dvf)
{0: Z, 1: C2, 2: 0}
```

The development of this integration is part of the tickets #30400, #30447, and #30862 in the SageMath Trac, which are currently under review by SageMath developers.

7. Conclusions and Further Work

In this paper we presented an interface between the computer algebra systems SageMath and Kenzo, together with an optional Kenzo package for SageMath. Our work has made it possible to work with Kenzo in a friendlier way and has allowed both systems

to collaborate in some computations which cannot be done independently in any of the programs. In addition, we enhanced SageMath with new functionalities in algebraic topology, allowing the representation and computation of topological invariants of spaces of an infinite nature; such type of spaces were not available in the system before. We have also integrated the first implemented algorithm to compute homotopy groups of not 1-reduced simplicial sets. This led us to compute homotopy groups that have never been determined before.

The first prototype of our interface and the Kenzo package, presented in [13], are already available from version 8.7 of SageMath and include some basic functionalities of algebraic topology computations. The new version, which has recently been accepted by SageMath developers and is available from SageMath 9.2, integrates some Kenzo external modules (which led us to find several bugs on one of them which have been fixed) and adds new functionalities and computations: Joins of two simplicial sets, wedges of spaces of infinite nature, smash products, homotopy groups, spectral sequences, and connection between chain complexes, simplicial sets, and morphisms of chain complexes. We also provide new computation capabilities about invariants of finite topological spaces; this part of the work is currently under review by the SageMath developers to be included in future releases.

The optional Kenzo package for SageMath and the interface to communicate both systems possess many interesting features:

- The package permits an easy installation of Kenzo and incorporates some of the external modules for Kenzo that are scattered in the personal websites of different authors. This has required the authors to update, adapt, polish, and correct flaws in the external code, since some of them were developed for outdated versions of the Kenzo system;
- They spread Kenzo to a broad community by including it in SageMath. The interface also eases the use of Kenzo for users with no previous knowledge of functional programming, since the interface is transparent: The user does not notice that Kenzo is running withing SageMath and the computations are carried out in Kenzo via SageMath commands. No knowledge on functional programming is required;
- Jupyter notebooks can be used as a friendly graphical user interface. Even more, it is possible to use SageMath with our Kenzo interface online with no software installation required;
- The interface is efficient due to the use of a C-library interface, roughly equivalent to Kenzo in time and space efficiency;
- They are extensible: New modules and features can be added to both the package and the interface;
- It is an open-source software: Any user can explore and adapt the code.

As possible further work, there exist other algebraic topology constructions which are implemented as additional modules in Kenzo and are not available in SageMath, such as discrete vector fields, generalized spectral sequences, or homology of group extensions. As done for the external modules presented in this paper, the integration of these constructions in SageMath requires one to adapt the Kenzo Common Lisp code to the Embeddable Common Lisp library in SageMath and to develop the corresponding wrappers and necessary functions in both Lisp and SageMath. Furthermore, other types of spectral sequences available in Kenzo could be integrated in SageMath and to do so, we plan to define in SageMath the notions of filtered complex and fibration (already available in Kenzo). To implement these notions as generally as possible, functional programming and lambda functions in SageMath will be necessary. Finally, following this idea of taking advantage of integrating several software packages, we would like to improve Kenzo by using libraries such as FLINT [36], which could be used for algebraic computations with matrices over rings necessary for homological computations.

Author Contributions: Conceptualization, J.C.-R., J.D., M.M.-B. and A.R.; Investigation, J.C.-R., J.D., M.M.-B. and A.R.; Methodology, J.C.-R., J.D., M.M.-B. and A.R.; Software, J.C.-R., J.D., M.M.-B. and A.R.; Writing—original draft, J.C.-R., J.D., M.M.-B. and A.R. All authors have read and agreed to the published version of the manuscript.

Funding: J. Cuevas-Rozo, J. Divasón, and A. Romero are partially supported by the Spanish Ministry of Science, Innovation and Universities, project MTM2017-88804-P. M. Marco-Buzunáriz is partially supported by MTM2016-76868-C2-2-P and Grupo "Álgebra y Geometría" of Gobierno de Aragón/ Fondo Social Europeo.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: We want to thank Francis Sergeraert, Gerd Heber, and Jónathan Heras for the development and maintenance of Kenzo and the external modules used in our work. We thank Carlos Alquézar who has helped us in developing automatic tests for our programs. We also want to thank John H. Palmieri and Travis Scrimshaw for their careful review on the ticket about the interface in the SageMath development Trac.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Bubenik, P. Statistical topological data analysis using persistence landscapes. *J. Mach. Learn. Res.* **2015**, *16*, 77–102.
2. Carlsson, G. Topology and data. *Bull. Am. Math. Soc.* **2009**, *46*, 255–308. [CrossRef]
3. Mischaikow, K.; Kokubu, H.; Mrozek, M.; Pilarczyk, P.; Gedeon, T.; Lessard, J.P.; Gameiro, M. The Computational HOMology Project. 2009. Available online: http://chomp.rutgers.edu/ (accessed on 24 March 2021).
4. Nanda, V. Perseus, the Persistent Homology Software. 2019. Available online: http://www.sas.upenn.edu/~vnanda/perseus (accessed on 24 March 2021).
5. Tausz, A.; Vejdemo-Johansson, M.; Adams, H. JavaPlex: A research software package for persistent (co)homology. In *Proceedings of ICMS 2014*; Hong, H., Yap, C., Eds.; Lecture Notes in Computer Science 8592; 2014; pp. 129–136. Available online: http://appliedtopology.github.io/javaplex/ (accessed on 24 March 2021).
6. Sergeraert, F. The computability problem in Algebraic Topology. *Adv. Math.* **1994**, *104*, 1–29. [CrossRef]
7. Dousson, X.; Rubio, J.; Sergeraert, F.; Siret, Y. The Kenzo Program. Institut Fourier, Grenoble. 1999. Available online: http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/ (accessed on 24 March 2021).
8. Whitehead, G. Fiber Spaces and the Eilenberg Homology Groups. *Proc. Natl. Acad. Sci. USA* **1952**, *38*, 426–430. [CrossRef] [PubMed]
9. Rubio, J.; Sergeraert, F. *Constructive Homological Algebra and Applications, Lecture Notes Summer School on Mathematics, Algorithms, and Proofs*; University of Genova: Genova, Italy, 2006. Available online: http://www-fourier.ujf-grenoble.fr/~sergerar/Papers/Genova-MAP-2006-v3.pdf (accessed on 24 March 2021).
10. Heras, J.; Pascual, V.; Rubio, J.; Sergeraert, F. fKenzo: A user interface for computations in Algebraic Topology. *J. Symb. Comput.* **2011**, *46*, 685–698. [CrossRef]
11. The SageMath Developers. SageMath, the Sage Mathematics Software System (Version 9.1.0). 2020. Available online: https://www.sagemath.org (accessed on 24 March 2021).
12. Palmieri, J. Examples of Simplicial Sets—Sage Reference Manual v8.8. 2019. Available online: https://doc.sagemath.org/html/en/reference/homology/sage/homology/simplicial_set_examples.html#sage.homology.simplicial_set_examples.simplicial_data_from_kenzo_output (accessed on 24 March 2021).
13. Cuevas-Rozo, J.; Marco-Buzunáriz, M.A.; Romero, A. Computing with Kenzo from Sage. In Proceedings of the 2019 conference on Effective Methods in Algebraic Geometry, 2019; p. 23. Available online: http://eventos.ucm.es/_files/_event/_12097/_editorFiles/file/Abstracts_MEGA-alphabetic.pdf (accessed on 24 March 2021).
14. Cuevas-Rozo, J.; Divasón, J.; Marco-Buzunáriz, M.; Romero, A. A Kenzo interface for algebraic topology computations in SageMath. *ACM Commun. Comput. Algebra* **2019**, *53*, 61–64. [CrossRef]
15. Rubio, J.; Sergeraert, F.; Siret, Y. *EAT: Symbolic Software for Effective Homology Computation*; Institut Fourier: Grenoble, France, 1997. Available online: https://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/EAT-program.zip (accessed on 24 March 2021).
16. Heber, G. A Repackaged Version of the Kenzo Program by Francis Sergeraert and Collaborators. 2019. Available online: https://github.com/gheber/kenzo (accessed on 24 March 2021).
17. May, J.P. *Simplicial Objects in Algebraic Topology*; Van Nostrand Mathematical Studies; University of Chicago Press: Chicago, IL, USA, 1967.

18. Heras, J. A Kenzo Module Computing Homotopy Groups. 2011. Available online: https://esus.unirioja.es/psycotrip/archivos_documentos/homotopy.cl (accessed on 24 March 2021).
19. Romero, A.; Rubio, J.; Sergeraert, F. Computing Spectral Sequences. *J. Symb. Comput.* **2006**, *41*, 1059–1079. [CrossRef]
20. Cuevas-Rozo, J. Finite Topological Spaces in Kenzo. 2020. Available online: https://github.com/jcuevas-rozo/finite-topological-spaces (accessed on 24 March 2021).
21. Marco, M.; Romero, A. Library Interface to Kenzo—Sage Reference Manual v9.2. 2020. Available online: https://doc.sagemath.org/html/en/reference/interfaces/sage/interfaces/kenzo.html (accessed on 24 March 2021).
22. Real, P. An algorithm computing homotopy groups. *Math. Comput. Simul.* **1996**, *42*, 461–465. [CrossRef]
23. Hurewicz, W. Beiträge zur Topologie der Deformationen I-II. *Proc. Akad. Van Wet.* **1935**, *38*, 112–119. 521–528.
24. Newman, M. *Integral Matrices*; Academic Press: Cambridge, MA, USA, 1972.
25. Cuevas-Rozo, J.; Divasón, J.; Marco-Buzunáriz, M.; Romero, A. A New Algorithm for Computing the Twisting Operator of the Whitehead Tower Method in the Not 1-Reduced Case. 2021. Available online: https://www.unirioja.es/cu/anromero/TwistingOperatorWhiteheadTower.pdf (accessed on 24 March 2021).
26. Babson, E.; Björner, A.; Linusson, S.; Shareshian, J.; Welker, V. Complexes Of Not i-Connected Graphs. *Topology* **1996**, *38*, 271–299. [CrossRef]
27. Heras, J. Effective Homology of the pushout of simplicial sets. In Proceedings of the XII Encuentros de Álgebra Computacional y Aplicaciones (EACA 2010), Santiago de Compostela, Spain, 19–21 July 2010; pp. 152–156.
28. McCleary, J. *A User's Guide to Spectral Sequences*, 2nd ed.; Number 58 in Cambridge Studies in Advanced Mathematics; Cambridge University Press: Cambridge, UK, 2001.
29. Serre, J.P. Homologie singulière des espaces fibrés. *Ann. Math.* **1951**, *54*, 425–505. [CrossRef]
30. Eilenberg, S.; Moore, J.C. Homology and fibrations, I: Coalgebras, cotensor product and its derived functors. *Comment. Math. Helv.* **1965**, *40*, 199–236. [CrossRef]
31. Alexandroff, P. Diskrete Räume. *Mat. Sb.* **1937**, *2*, 501–518.
32. Stong, R.E. Finite topological spaces. *Trans. Am. Math. Soc.* **1966**, *123*, 325–340. [CrossRef]
33. Cuevas-Rozo, J.; Lambán, L.; Romero, A.; Sarria, H. Effective homological computations on finite topological spaces. *Appl. Algebra Eng. Commun. Comput.* 2020, In press. [CrossRef]
34. Jipsen, P.; Saliola, F. Finite Posets—Sage Reference Manual v9.2. 2020. Available online: https://doc.sagemath.org/html/en/reference/combinat/sage/combinat/posets/posets.html (accessed on 24 March 2021).
35. Barmak, J.A. *Algebraic Topology of Finite Topological Spaces and Applications*; *Lecture Notes in Mathematics*; Springer: Berlin/Heidelberg, Germany, 2011; Volume 2032.
36. Hart, W.; Johansson, F.; Pancratz, S. FLINT: Fast Library for Number Theory, Version 2.4.0. 2013. Available online: http://flintlib.org (accessed on 24 March 2021).