1

Preview

Homology is a very powerful tool in that it allows one to draw conclusions about *global* properties of spaces and maps from *local* computations. It also involves a wonderful mixture of algebra, combinatorics, computation, and topology. Each of these subjects is, of course, interesting in its own right and appears as the subject of multiple sections in this book. But our primary objective is to see how they can be combined to produce homology, how homology can be computed efficiently, and how homology provides us with information about the geometry and topology of nonlinear objects and functions. Given the amount of theory that needs to be developed, it is easy to lose sight of these objectives along the way.

Therefore, we begin with a preview. We start by considering applications of homology and then provide a heuristic introduction to the underlying mathematical ideas of the subject. The point of this chapter is to provide intuition for the big picture. As such we will, without explanation, introduce some fundamental terminology with the expectation that the reader will remember the words. Precise definitions and mathematical details are provided in the rest of the book.

1.1 Analyzing Images

It is hard to think of a scientific or engineering discipline that does not generate computational simulations or make use of recording devices or sensors to produce or collect image data. It is trivial to record simple color videos of events, but it should be noted that such a recording can easily require about 25 megabytes of data per second. While obviously more difficult, it is possible, using X-ray computed tomography, to visualize cardiovascular tissue with a resolution on the order of 10 μm. Because this can be done at a high speed, timed sequences of three-dimensional images can be constructed. This technique can be used to obtain detailed information about the geometry and function of the heart, but it entails large amounts of data. Obviously, the size

and complexity of this data will grow as the sophistication of the sensors or simulations increases.

These large amounts of data are a mixed blessing; while we can be more confident that the desired information is captured, extracting the relevant information in a sea of data can become more difficult. One solution is to develop automated methods for image processing. These techniques are often, if somewhat artificially, separated into two categories: *low-level vision* and *high-level vision*. Typical examples of the latter include object recognition, optical character and handwriting recognizers, and robotic control by visual feedback. Low-level vision, on the other hand, focuses on the geometric structures of the objects being imaged. As such it often is a first step toward higher-level tasks. It is our belief that computational homology has the potential to play an important role in low-level vision. Notice the phrasing of this last sentence: The use of algebraic topology in imaging is, at the time that this book is being written, a very new subject. We hope that this work will open doors to exciting endeavors.

Let us begin by using some extremely simple figures to get a feel for what homology measures. Consider the line segments in Figure 1.1(a) and (b). For a topologist the most important distinguishing properties of these figures is not that they are made up of straight line segments, nor that the lengths of the line segments are different, but rather that in Figure 1.1(a) we have an object that consists of one *connected component* and in Figure 1.1(b) the object consists of two distinct pieces. Homology provides us with a means of measuring this. In particular, if we call the object in Figure 1.1(a) X and the object in Figure 1.1(b) Y, then the *zeroth homology groups* of these figures are

$$H_0(X) \cong \mathbf{Z} \quad \text{and} \quad H_0(Y) \cong \mathbf{Z}^2.$$

We will explain later what this means or how it is computed. For the moment it is sufficient to know that homology allows us to assign to a topological space (e.g., X or Y) an *abelian group* and that the dimension of this group counts the number of distinct pieces that make up the space.

(a) (b)

Fig. 1.1. (a) A line segment consisting of one connected component. (b) A divided line segment consisting of two connected components.

Given that there is a zeroth homology group, the reader might suspect that there is also a *first homology group*. This is correct and it measures different topological information. Again, let us consider some simple figures. In Figure 1.2(a) we see a line segment in the plane. If we think of this as a

piece of fencing, it is clear that it does not enclose any region in the plane. On the other hand, Figure 1.2(b) clearly encloses the square $(0,1) \times (0,1)$. If we add more segments we can enclose more squares as in Figure 1.2(c), though, of course, some segments need not enclose any region. Finally, as indicated by the shaded square in Figure 1.2(d), by filling in some regions we can eliminate enclosed areas.

The first homology group measures the number of these enclosed regions and for each of these figures, denoted respectively by X_a, X_b, X_c, and X_d, it is as follows:

$$H_1(X_a) = 0, \quad H_1(X_b), \cong \mathbf{Z}, \quad H_1(X_c), \cong \mathbf{Z}^2, \quad \text{and} \quad H_1(X_d), \cong \mathbf{Z}.$$

Notice that even though the drawings in Figure 1.2(b)–(d) are more elaborate than those of Figure 1.1, the number of connected components is always one, thus

$$H_0(X_i) \cong \mathbf{Z}, \qquad i = a, b, c, d.$$

One final, but extremely important, comment is that the homology of an object does not depend on the ambient space. Hence if we were to lift the objects of Figure 1.2 from the plane and think of them as existing in $\mathbf{R}^3$ or in any abstract higher-dimensional space, their homology groups would not change. After reading this book the reader will realize that the homologies of such different things as a garden hose and a coffee mug are the same as the homology of X_b. Actually, homology is not measuring size or even a specific shape; rather it measures very fundamental properties like the number of holes and pieces. We will return to this point shortly.

If it were not for the fact that we introduce the words "homology group" in our discussion of Figures 1.1 and 1.2, the previous comments would be completely trivial. So let us try to rephrase the statements in the form of a nontrivial question.

> *Can we develop a computationally efficient algebraic tool that tells us how many connected components and enclosed regions a geometric object contains?*

For example, it would be nice to be able to enter the mazelike object of Figure 1.3 into a computer and have the computer tell us whether the figure consisting of all black line segments is connected and whether there are one or more enclosed white regions in the figure. By the end of Chapter 4 the reader will be able to do this and much more.

This last suggestion raises a fundamental question: What does it mean to enter a figure into the computer? Consider, for example, the left-hand image of Figure 1.4, which shows two circles that intersect. There are a variety of ways in which we could attempt to represent the circles. In a mathematics class they would typically be understood to be smooth curves, composed of uncountably many points. However, this picture was produced by a computer, and thus only a finite amount of information is presented. The right-hand image of

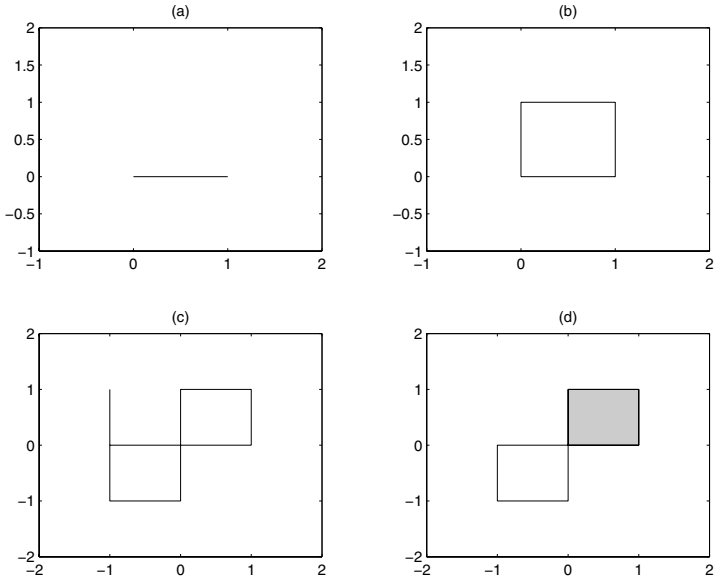


Fig. 1.2. (a) A simple line segment in the plane does not enclose any region. (b) These four line segments enclose the region $(0,1) \times (0,1)$. (c) It is easy to bound more than one region. (d) By filling in a region we can eliminate enclosed regions.

Figure 1.4 is obtained by zooming in on the upper point of the intersection of the circles. Observe that the smooth curves have become chains of small squares. These squares represent the smallest geometric units of information presented in the image on the left-hand side.

Given our goal of an automated method for image processing, it seems that these squares are a natural input for the computer. Hence we could enter the circles in Figure 1.4 into the computer by listing the black squares.

More interesting images are, of course, more complicated. Consider the photo in Figure 1.5 of the moon's surface in the Sea of Tranquillity taken from the Apollo 10 spacecraft. This is a black-and-white photo that, if rendered on a computer screen, would be presented as a rectangular set of elements each one of which is called a picture element, or a *pixel* for short. Each pixel has a specific light intensity determined by an integer gray-scale value between 0 and 255. This rendering captures the essential elements of a digital image: The image must be defined on a discretized domain (the array of pixels), and similarly the observed values of the image must lie in a discrete set (gray-scale values).

On the other hand, the Sea of Tranquillity is an analog rather than a digital object. Its physical presence is continuous in space and time. Furthermore, the visual intensity of the image that we would see if we were observing the moon directly also takes a continuum of values. Clearly, information is being lost

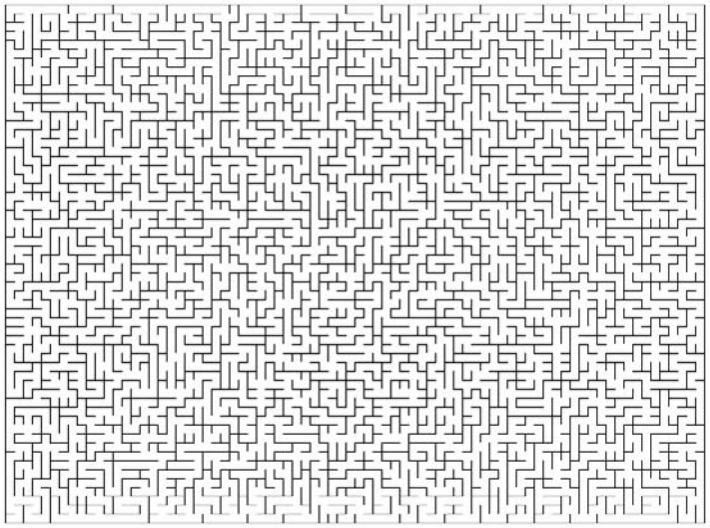


Fig. 1.3. How many distinct bounded regions are in this complicated maze?

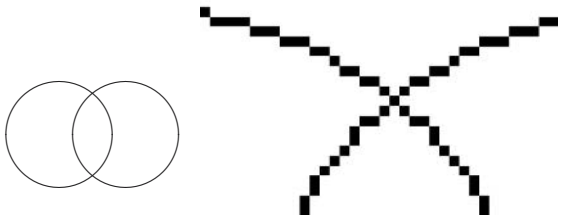


Fig. 1.4. On the left we have a simple image of two intersecting circles produced by a computer. On the right is a magnification of the upper point at which the circles intersect. Observe that the smooth curve is now a chain of squares that intersect either at a vertex or on a face.

in the process of describing an analog object in terms of digital information. This is an important point and is the focus of considerable research in the image processing community. However, these problems lie outside the scope of this book and, with the exception of Section 8.1 and occasional comments, will not be discussed further.

On a more positive note, we can use this simple example to indicate the value of being able to count "holes." One of the more striking features of

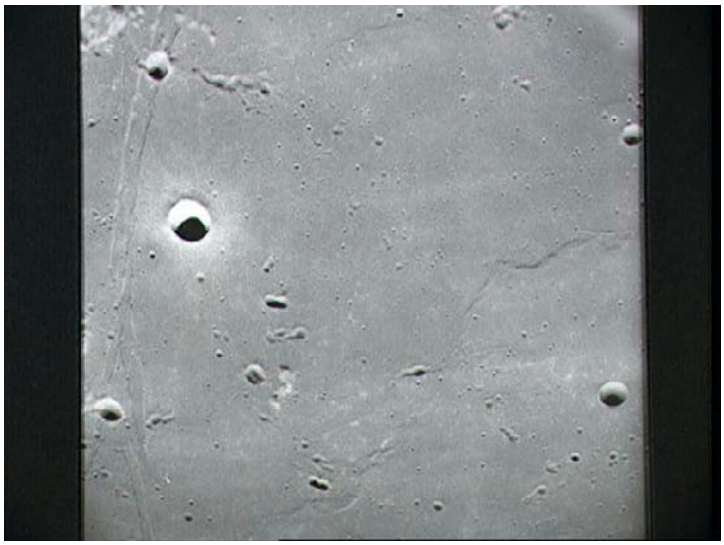


Fig. 1.5. Near-vertical photograph taken from the Apollo 10 Command and Service Modules shows features typical of the Sea of Tranquillity near Apollo Landing Site 2. The original is a 70-mm black-and-white photo (see the NASA web page http://images.jsc.nasa.gov/iams/images/pao/AS10/10075149.jpg).

Figure 1.5 is the craters and the natural question is: How many are there? To answer this we first need to decide which pixels represent the smooth surface and which represent the cratered surface. Thus we want to reduce this picture to a binary image that distinguishes between smooth and cratered surface areas.

The simplest approach for reducing a gray-scale image to a binary image is *image thresholding*. One chooses a range of gray-scale values $[T_0, T_1]$; all pixels whose values lie in $[T_0, T_1]$ are colored black while the others are left white. Of course, the resulting binary image depends greatly on the values of T_0 and T_1 that are chosen. Again, the question of the optimal choice of $[T_0, T_1]$ and the development of more sophisticated reduction techniques are serious problems that are not dealt with in this book.

Having acknowledged the simplistic approach we are adopting here, consider Figure 1.6. These binary images were obtained from Figure 1.5 as follows. Recall that in a gray-scale image each pixel has a value between 0 and 255, where 0 is black and 255 is white. The craters are darker in color (which corresponds to a lower gray-scale value). Since there is no absolute definition of which gray scales correspond to which craters, we choose two threshold intervals: $[95, 255]$ and $[105, 255]$. Pixels in theses ranges are taken to represent the smooth surface: the black pixels in Figure 1.6 correspond to the lightest pixels in Figure 1.5 and the white pixels to the darkest. In other words, the black

region is indicative of the smooth surface and the white regions are craters. It should not come as a surprise that different thresholding intervals result in different binary images.

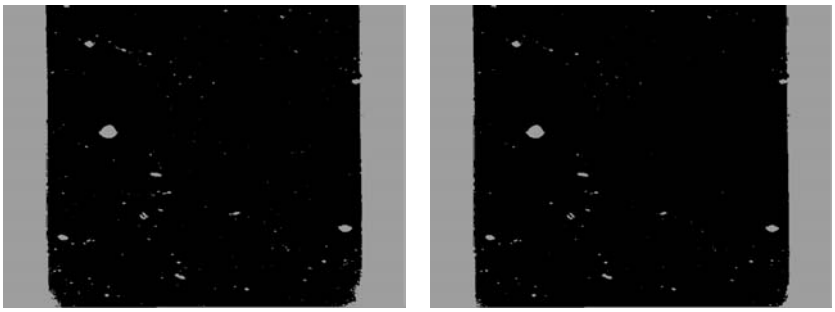


Fig. 1.6. The left figure was obtained from Figure 1.5 by choosing the threshold interval $[95, 255]$ while the right figure corresponds to the threshold interval $[105, 225]$.

Counting the number of holes (white regions bounded by black pixels) in the binary pictures provides an approximation of the number of craters in the picture. Observe that we are back to the problem motivated by Figures 1.2 and 1.3. Thus we want to be able to compute the first homology group for the object defined by the black pixels.

The examples presented so far, namely Figures 1.2 and 1.3 and the lunar photograph, were included—under the assumption that a picture saves a thousand words—to help develop intuition. The full mathematical machinery of homology is not necessary to analyze such simple images in the plane, and we do not recommend the material in this book for a reader whose only interest is in counting craters on the moon. For example, we could have identified the craters by choosing to threshold with the intervals $[0, 95]$ and $[0, 105]$ and then counting the number of connected pieces. This latter task requires no knowledge of homology. On the other hand, many physical problems are higher-dimensional, where our visual intuition fails and topological reasoning becomes crucial.

To choose a specific problem where computational topology has been employed, we turn to the subject of metallurgy and in particular to the work of [53, 39]. Consider a binary alloy consisting of iron (Fe) and chromium (Cr) created by heating the metals to a high temperature. The initial configuration of the iron and chromium atoms is essentially spatially homogeneous, up to small, random variations. However, upon cooling, the iron and chromium atoms separate, leading to a two-phase microstructure; that is, the material divides into regions that consist primarily of iron atoms or chromium atoms, but not both. These regions, which we will denote by $F(t)$ for iron and $C(t)$ for chromium, are obviously three-dimensional structures, can be extremely

complicated, and furthermore, change their form with time t. Current technology allows for accurate three-dimensional measurements to be performed on the atomic level (essentially the material is serially sectioned and then examined with an atomic probe; see [53] for details). Thus these regions can be experimentally determined. Of course, for each sample the actual geometric structure of $F(t)$ and $C(t)$ will be different since the initial configurations of iron and chromium atoms are distinct. On the theoretical side there are mathematical models meant to describe this process of decomposition. The easiest to state mathematically takes the form of the Cahn–Hilliard equation

$$\frac{\partial u}{\partial t} = -\Delta(\epsilon^2 \Delta u + u - u^3), \quad x \in \Omega, \tag{1.1}$$

$$n \cdot \nabla u = n \cdot \nabla \Delta u = 0, \quad x \in \partial\Omega, \tag{1.2}$$

where n is the outward normal to $\partial\Omega$. Of particular interest is the case where $\epsilon > 0$ but small, since under this condition solutions to this equation produce complicated patterns. For example, Figure 1.7 contains a plot of the level set S defined by $u(x, y, z, \tau) = 0$ on the domain $\Omega = [0, 1]^3$ where $\epsilon = 0.1$. This was obtained by starting with a small but random initial condition $u_0(x, y, z)$ satisfying

$$\int\int\int_{\Omega} u_0(x, y, z)\, dx\, dy\, dz = 0.$$

Since (1.1) is a nonlinear partial differential equation, there is no hope of obtaining an analytic formula for the solution. Thus $u(x, y, z, \tau)$ was computed numerically on a grid consisting of $128 \times 128 \times 128$ cubical elements until time $t = \tau$. This means that $u(x, y, z, \tau)$ is approximated by a set of numbers $\{u(i, j, k, \tau) \mid 1 \le i, j, k \le 128\}$.

Returning to the issue of modelling alloys, the assumption is that positive values of u indicate higher density of one element and that negative values of u indicate higher density of the other element. More precisely, for some $\delta > 0$ but small, the region of one phase is given by $R_1(t) := \{x \in \Omega \mid u(x, t) > 1 - \delta\}$ and the other by $R_2(t) := \{x \in \Omega \mid u(x, t) < -1 + \delta\}$ (see [71] for a broad introduction to models of pattern formation). With appropriate modifications to (1.1) (see [53] for the details), one can then numerically simulate the iron–chromium alloy; in particular, one can try to compare $F(t)$ with $R_1(t)$ and $C(t)$ with $R_2(t)$.

At this point this problem's need for an algebraic measure of the topological structure can be made clear. Recall that in the material one starts with an essentially random configuration of iron and chromium atoms. To model this numerically, one begins with a random initial condition $u(x, 0)$ where $|u(x, 0)| < \mu$ for all $x \in \Omega$ and μ is small. Equation (1.1) is then solved until $t = t_0$. We now wish to compare $F(t_0)$ against $R_1(t_0)$ and $C(t_0)$ against $R_2(t_0)$. However, since the initial conditions are random, it makes no sense to demand that $F(t_0) = R_1(t_0)$ or even that they be close to one another. Instead we ask if they are similar in the sense of their topology. More precisely,

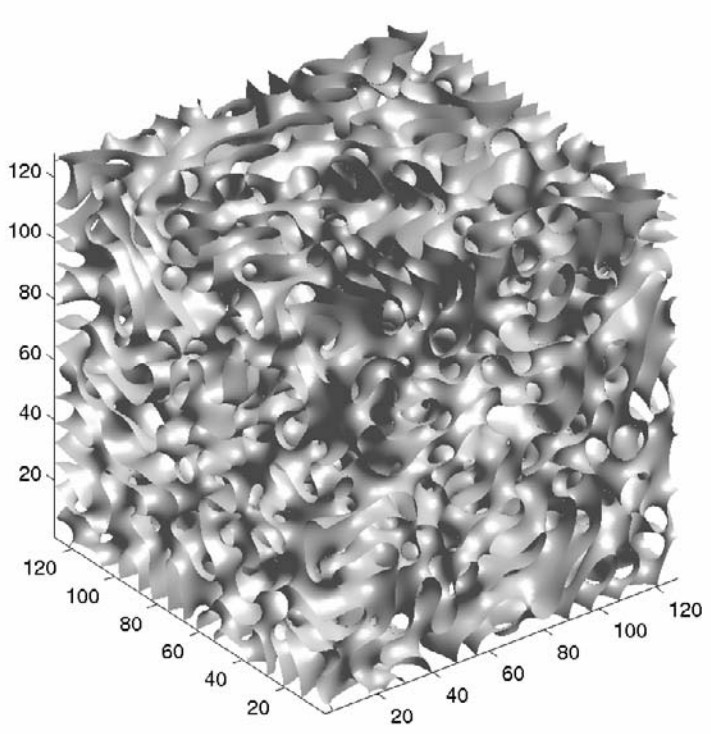


Fig. 1.7. A graphical rendering of the level set S defined by $u(x, y, z, \tau) = 0$. This was obtained by starting with a small but random initial condition on a grid consisting of $128 \times 128 \times 128$ elements and using a finite-element method numerically solving (1.1) until time $t = \tau$. It should be noted that the object is constructed by means of a triangulated surface using the data points $\{u(i, j, k, \tau) \mid 1 \leq i, j, k \leq 128\}$. This is a standard procedure in computer graphics as it produces an object that is much more pleasant to view. The homology groups for the triangulated surface S are $H_0(S) = \mathbf{Z}$, $H_1(S) = \mathbf{Z}^{1701}$, and $H_2(S) = 0$.

each of the regions $F(t_0)$, $C(t_0)$, or $R_i(t_0)$ can be made up of a multitude of components. There can be tunnels that pass through the regions and even hollow cavities.

We have suggested that the zeroth and first homology groups can be used to identify the number of components and tunnels. Thus we could try to compare

$$H_0(F(t)) \text{ versus } H_0(R_1(t)) \quad \text{and} \quad H_0(C(t)) \text{ versus } H_0(R_2(t))$$

and

$$H_1(F(t)) \text{ versus } H_1(R_1(t)) \quad \text{and} \quad H_1(C(t)) \text{ versus } H_1(R_2(t)).$$

As we saw with the examples of Figure 1.2, even if the objects are not identical in shape they can have the same homology groups.

Of course, these comparisons ignore the question of cavities. For this we need the *second homology group*; that is, we should compare

$$H_2(F(t)) \text{ versus } H_2(R_1(t)) \quad \text{and} \quad H_2(C(t)) \text{ versus } H_2(R_2(t)).$$

These types of comparisons are performed in [39]. However, the reader who consults [39] will not find the words "homology group" in the text. There are probably two reasons for this. First, this vocabulary is not common knowledge in the metallurgy community. Second, and more importantly, because the regions are three-dimensional, computational tricks can be employed to circumvent the need to explicitly compute the homology groups.

The intention of the last sentence by no means is to suggest that the reader who is interested in these kinds of problems can avoid learning homology theory. The tricks need to be justified, and the simplest justification makes essential use of algebraic topology. This dichotomy between the algebraic theory and the computational methods will be made clear in this book. After all, our goal is not only to compute homology groups, but to do so in an efficient manner. Thus in Chapter 3 we provide a purely algebraic algorithm for computing homology. This guarantees that homology groups are always computable. However, this method is extremely inefficient. Thus in Chapter 4 we introduce reduction algorithms that are combinatorial in nature and reasonably fast. Nevertheless, the justification of these latter algorithms depends crucially on a solid understanding of the algebraic theory.

In the next example, which is essentially four-dimensional, the tricks employed by [39] no longer work, and even on the level of language we can begin to appreciate the advantage of an abstract algebraic approach to the topic.

Figure 1.8 is a tomographic image of a horizontal slice of a human heart. Depending on the machine, several such images representing different cross sections can be taken simultaneously. Combining these two-dimensional images results in a three-dimensional image made up of three-dimensional cubes or *voxels*. As in the case of the lunar photo, a gray scale is assigned to each voxel. Assume that by using appropriate thresholding techniques we can identify those voxels that correspond to heart tissue. Then, using the language of the previous example, cavities could be identified with chambers and tunnels might indicate blood vessels, valves, or even defects such as holes in the heart.

However, medical technology allows us to go further. Multiple images can be obtained within the time span of a single heartbeat. Thus the full data set results in a four-dimensional object—three space dimensions and one time dimension—where the individual data elements are four-dimensional cubes or *tetrapus*. At this stage standard English begins to fail. As a simple example, consider a chamber of the heart at an instant of time when the valves are closed. In this case, the chamber is a cavity in a three-dimensional object. However, if we include time, then the valve will open and close so the three-dimensional cavity does not lead to a four-dimensional cavity. Which of the

homology groups characterizes such a region? By the end of Chapter 2 the reader will know the answer.

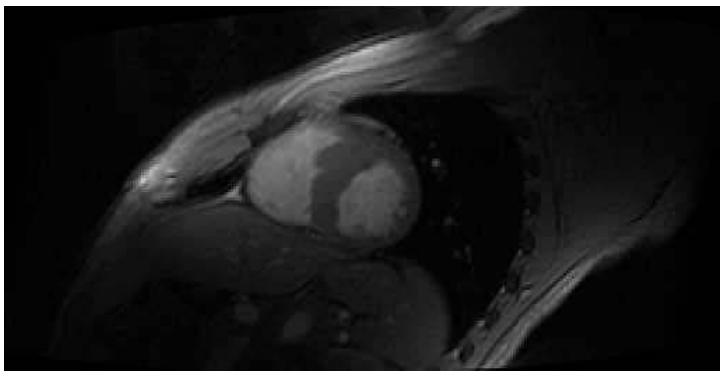


Fig. 1.8. A tomographic section of a human heart produced by the Surgical Planning Lab, Department of Radiology, Brigham and Women's Hospital.

1.2 Nonlinear Dynamics

In the previous section we present examples that suggest the need for algorithms to analyze the geometric structure of objects. We now turn to problems where it is important to have algorithms for studying nonlinear functions.

As motivation we turn to a simple model for population dynamics. Let y_n represent a population at time n. The simplest possible assumption is that y_{n+1}, the population one time unit later, is proportional to y_n. In this case $y_{n+1} = ry_n$, where r is the growth rate. An obvious problem with this model is that when $r > 1$, the population can grow to arbitrary sizes, which cannot happen, because the resources are limited. This issue can be avoided by assuming that the rate of growth is a decreasing function of the population. For simplicity let $r(y) = K - y$ for some constant K. Then the population at time $n + 1$ is given by

$$y_{n+1} = r(y_n)y_n = (K - y_n)y_n. \tag{1.3}$$

If we use the change of variables $x = y/K$, (1.3) takes the form

$$x_{n+1} = Kx_n(1 - x_n).$$

Notice that the new variable x_n represents the scaled population level at time n.

To simplify the discussion, let us choose $K = 4$ and write

$$x_{n+1} = f(x_n) = 4x_n(1 - x_n).$$

A natural question is: Given an initial population level x_0, what are the future levels of the population? Notice that producing the sequence of population levels

$$x_0, x_1, x_2, \ldots, x_n, \ldots$$

is equivalent to iterating the function f. Clearly, one can write a simple program that performs such an operation. However, before doing so we wish to make two observations. First, because we are talking about populations, we are only interested in $x \geq 0$. Thus the model obviously has some flaws since if $x_0 > 1$, then $x_1 = f(x_0) < 0$. For this reason we will assume that $0 \leq x_0 \leq 1$. Second, $f([0, 1]) = [0, 1]$. So if we begin with the restricted initial conditions, then our population always remains nonnegative.

Figure 1.9 shows a sequence of populations $\{x_i \mid i = 0, \ldots, 100\}$ where $x_0 = 0.1$. One of the most striking features of this plot is the lack of a specific pattern. This raises a series of ever more difficult questions.

1. Do initial conditions exist for which the population is fixed, that is, $x_0 = x_1 = x_2 = \ldots$? Observe that this is equivalent to asking if there is a solution to the equation $f(x) = x$.
2. Do initial conditions exist that lead to *periodic orbits* of a given period? More specifically, given a positive integer k, does an initial condition x_0 exist such that $f^k(x_0) = x_0$ but $f^j(x_0) \neq x_0$ for all $j = 1, 2, \ldots, k - 1$? In this case we would say that we have found a periodic orbit with *minimal period* k.
3. What is the set of all k for which there exists a periodic orbit with minimal period k? How many such orbits are there?
4. Are there many orbits that, like that of Figure 1.9, seem to have no predictable pattern? Are there any simple rules that such orbits must satisfy?

For this map the answer to the first question is obvious:

$$f(x) = x \quad \text{if and only if} \quad x = 0 \text{ or } x = \frac{3}{4}.$$

The astute reader will realize that, for any fixed k, finding a periodic orbit of that period is the same as solving $f^k(x) - x = 0$. But finding explicit solutions to $f^k(x) - x$ for $k > 2$ is a difficult problem. Moreover, in many applications one is forced to deal with a map $f : X \to X$, where either X is a potentially high-dimensional and/or complicated space, or f is not known explicitly; for example, f is the time one map of a differential equation and can be found only by numerical integration.

For this particular map, answers to even the third and fourth questions are reasonably well understood [73]. However, given a particular function $f : \mathbf{R}^n \to \mathbf{R}^n$, for $n \geq 2$, our knowledge of the dynamics of f most likely comes from numerical simulations. With this in mind, let us return to the numerically computed orbit of Figure 1.9 and ask ourselves if we can trust

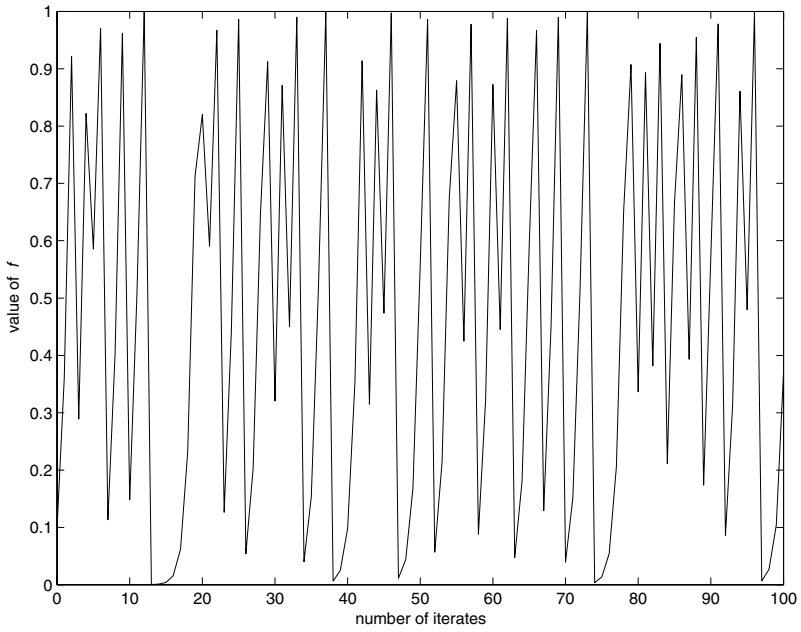


Fig. 1.9. A computed orbit for the logistic map $f(x) = 4x(1-x)$ with initial condition $x_0 = 0.1$.

the computation. Figure 1.10 shows two sequences of population levels. The circles indicate the sequence $\{x_i \mid i = 0, \ldots, 15\}$, where $x_0 = 0.1000$ and the stars represent $\{y_i \mid i = 0, \ldots, 15\}$, where $y_0 = 0.1001$. Observe that after 10 steps there is little correlation between the two sequences. In fact,

$$|x_{13} - y_{13}| \geq 0.9515.$$

Since the trajectories are forced to remain in the interval $[0, 1]$, this effectively states that a mistake on the order of 10^{-4} leads to an error that is essentially the same size as the entire range of possible values. This kind of phenomenon is often referred to as chaotic dynamics. Since numerical computations induce errors merely by the fact that the computer is incapable of representing numbers to infinite precision, in a chaotic system any single computed trajectory is suspect.

Hopefully this discussion has demonstrated that numerical simulations of dynamical systems need to be treated with respect. What is probably not clear is how computational homology can be used. A precise answer is the subject of Chapter 10. For the moment we will have to settle for some suggestive comments.

Let $f : \mathbf{R} \to \mathbf{R}$ be continuous and let us try to determine if f has a fixed point, that is, if there is a solution to the problem $f(x) = x$. Observe that

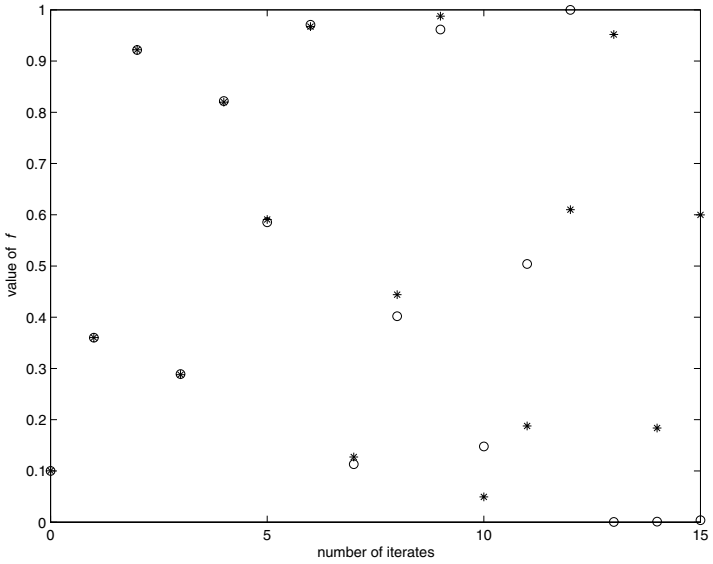


Fig. 1.10. Two computed orbits for the logistic map $f(x) = 4x(1-x)$. The $\circ$ and $*$ correspond to initial conditions $x_0 = 0.1$ and $y_0 = 0.1001$, respectively.

this is equivalent to showing that $f(x) - x = 0$. Let $g(x) = f(x) - x$, and assume that we determine that $g(a) < 0$ and $g(b) > 0$ for some $a < b$. By the intermediate value theorem, there exists $c \in (a, b)$ such that $g(c) = 0$ and hence $f(c) = c$.

Observe that a key element in this approach is the assumption that f and hence g are continuous. This is a topological assumption. Also notice that we do not need to know $g(a)$ nor $g(b)$ precisely; it is sufficient to show that the inequalities are satisfied. We know that the typical numerical computation will result in errors, thus a result of this form is encouraging. Therefore, let us try to be a little more precise.

Given an input x, we would like to have the output $g(x)$. However, the computer produces a potentially different value, which we will denote by $g_{num}(x)$. Assume that we can obtain an error bound for g_{num}, that is a number μ such that $|g(x) - g_{num}(x)| < \mu$. Returning to the fixed-point problem; if the computer determines that $g_{num}(a) < -\mu$ and $g_{num}(b) > \mu$, then we know that $g(a) < 0$ and $g(b) > 0$ and hence we can conclude the existence of a fixed point.

Generalizing this result to higher dimensions is not trivial. In the previous section we promise that Chapter 2 will show how homology groups are generated by topological spaces. In Chapter 6 we will go a step further and show that given a continuous map $f : X \to Y$ between topological spaces, there are unique linear maps $f_{*k} : H_k(X) \to H_k(Y)$, called the *homology maps*,

from the homology groups of one space to the homology groups of the other. Furthermore, we will show that to correctly compute the homology maps we do not need to know the function f explicitly, but rather it is sufficient to know a numerical approximation, f_{num}, and an appropriate error bound μ.

One of the most remarkable theorems involving homology is the Lefschetz fixed-point theorem (see Theorem 10.46), which guarantees the existence of a fixed point if the traces of the homology maps satisfy a simple condition. Since even with numerical error we can compute these homology maps correctly, this allows us to use the computer to give mathematically rigorous arguments for the existence of fixed points for high-dimensional nonlinear functions. In fact, at the end of Chapter 10 we show that generalizations of these types of arguments can be used to rigorously answer any of the four questions posed at the beginning of this section.

1.3 Graphs

The previous two sections are meant to be an enticement, suggesting the power and potential for homology in a variety of applications. But no attempt is made to explain how or why there should be an algebraic theory that can perform the needed measurements. We hope to rectify this, at least on a heuristic level, in the next few sections. Since we are still trying to motivate the subject, we will keep things as simple as possible. In particular, let us stick to objects such as those of Figures 1.2 and 1.3.[1]

To do mathematics we need to make sure that these simple objects are well defined. Graphs provide a nice starting point.

Definition 1.1 A *graph* G is a subset of $\mathbf{R}^3$ made up of a finite collection of points $\{v_1, \ldots, v_n\}$, called *vertices*, together with straight-line segments $\{e_1, \ldots, e_m\}$, joining vertices, called *edges*, which satisfy the following intersection conditions:

1. The intersection of distinct edges either is empty or consists of exactly one vertex, and
2. if an edge and a vertex intersect, then the vertex is an endpoint of the edge.

More explicitly, an edge joining vertices v_0 and v_1 is the set of points

$$\{x \in \mathbf{R}^3 \mid x = tv_0 + (1-t)v_1,\ 0 \leq t \leq 1\},$$

which is denoted by $[v_0, v_1]$.

A *path* in G is an ordered sequence of edges of the form

[1] We temporarily ignore the previously discussed two-dimensional pixel structure visible when zooming in on computer-generated figures and we view them here as one-dimensional objects composed of line segments.

$$\{[v_0, v_1], [v_1, v_2], \ldots, [v_{l-1}, v_l]\}.$$

This path *begins* at v_0 and *ends* at v_l. Its *length* is l, the number of its edges. A graph G is *connected* if, for every pair of vertices $u, w \in G$, there is a path in G that begins at u and ends at w. A *loop* is a path that consists of distinct edges and begins and ends at the same vertex. A connected graph that contains no loops is a *tree*.

Observe that Figures 1.2(a), (b), c) and Figure 1.3 are graphs. Admittedly they are drawn as subsets of $\mathbf{R}^2$, but we can think of $\mathbf{R}^2 \subset \mathbf{R}^3$. Figure 1.2(d) is not a graph. It is also easy to see that Figures 1.2(b) and (c) contain loops, and that Figure 1.2(a) is a tree. What about Figure 1.3?

Graphs, and more generally topological spaces, *cannot* be entered into a computer. As defined above a graph is a subset of $\mathbf{R}^3$ and so consists of infinitely many points, but a computer can only store a finite amount of data. Of course, there is a very natural way to *represent* a graph, which only involves a finite amount of information.

Definition 1.2 A *combinatorial graph* is a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a finite set whose elements are called *vertices* and $\mathcal{E}$ is a collection of pairs of distinct elements of $\mathcal{V}$ called *edges*. If an edge e of a combinatorial graph consists of the pair of vertices v_0 and v_1, we will write $e = [v_0, v_1]$.

It may seem at this point that we are making a big deal out of a minor issue. Consider, however, the sphere, $\{(x, y, z) \in \mathbf{R}^3 \mid x^2 + y^2 + z^2 = 1\}$. It is not so clear how to represent this in terms of a finite amount of data, but as we shall eventually see, we can compute its homology via a combinatorial representation. In fact, homology is a combinatorial object; this is what makes it such a powerful tool.

Even in the setting of graphs the issue of combinatorial representations is far from clear. Consider, for example, the graph $G = [0, 1] \subset \mathbf{R}$. How should we represent it as a combinatorial graph? Since we have not said what the vertices and edges are, the most obvious answer is to let $\mathcal{V} = \{0, 1\}$ and $\mathcal{E} = \{[0, 1]\}$. However, G could also be thought of as the graph containing the vertices 0, $1/2$, 1 and the edges $[0, 1/2]$, $[1/2, 1]$, in which case the natural combinatorial representation is given by $\mathcal{V}_2 = \{0, 1/2, 1\}$ and $\mathcal{E}_2 = \{[0, 1/2], [1/2, 1]\}$. More generally, we can think of G as being made up of many short segments leading to the combinatorial representation

$$\mathcal{V}_n := \{j/n \mid j = 0, \ldots, n\}, \quad \mathcal{E}_n := \{[j/n, (j+1)/n] \mid j = 0, \ldots, n-1\}.$$

We are motivating homology in terms of graphs (i.e., subsets of $\mathbf{R}^3$), however, the input data to the computer is in the form of a combinatorial graph, namely a finite list. Thus, to prove that homology is an invariant of a graph, we have to show that given any two combinatorial graphs that represent the same set, the corresponding homology is the same. This is not trivial! In fact, it will not be proven until Chapter 6.

On the other hand, in this chapter we are not supposed to be worrying about details, so we won't. However, we should not forget that there is this potential problem:

Can we make sure that two different combinatorial objects that give rise to the same set also give rise to the same homology?

Before turning to the algebra, we want to prove a simple property about trees.

A vertex that only intersects a single edge is called a *free vertex*.

Proposition 1.3 *Every tree contains at least one free vertex.*

Proof. Assume not. Then there exists a tree T with 0 free vertices. Let n be the number of edges in T. Let e_1 be an edge in T. Label its vertices by v_1^- and v_1^+. Since T has no free vertices, there is an edge e_2 with vertices $v_2^\pm$ such that $v_1^+ = v_2^-$. Continuing in this manner we can label the edges by e_i and the vertices by $v_i^\pm$, where $v_i^- = v_{i-1}^+$. Since there are only a finite number of vertices, at some point in this procedure we get $v_i^+ = v_j^-$ for some $i > j \geq 1$. Then $\{e_j, e_{j+1}, \ldots, e_i\}$ forms a loop. This is a contradiction. □

Exercises

1.1 Associate combinatorial graphs to Figures 1.2(a), (b), and (c).

1.2 Among all paths joining two vertices v and w at least one has minimal length. Such a path is called minimal. Show that any two edges and vertices on a minimal path are different.

1.3 Let T be a tree with n edges. Prove that T has $n + 1$ vertices.
Hint: Argue by induction on the number of edges.

1.4 Topological and Algebraic Boundaries

As stated earlier, our goal is to develop an algebraic means of detecting whether a set bounds a region or not. So we begin with the two simple sets of Figure 1.11, an interval I and the perimeter I^1 of a square. We want to think of these sets as graphs. As is indicated in Figure 1.11, we represent both sets by graphs consisting of four edges. The difference lies in the set of vertices.

We mentioned earlier that homology has the remarkable property that local calculations lead to knowledge about global properties. As already observed, the difference between the combinatorial graphs of I and I^1 is found in the vertices, which are clearly local objects. So let us focus on vertices and observe that they represent the endpoints or, as we shall call them from now on, the boundary points of edges.

Consider both the graph and the combinatorial graph of I. The left-hand column of Table 1.1 indicates the boundary points of each of the edges. The

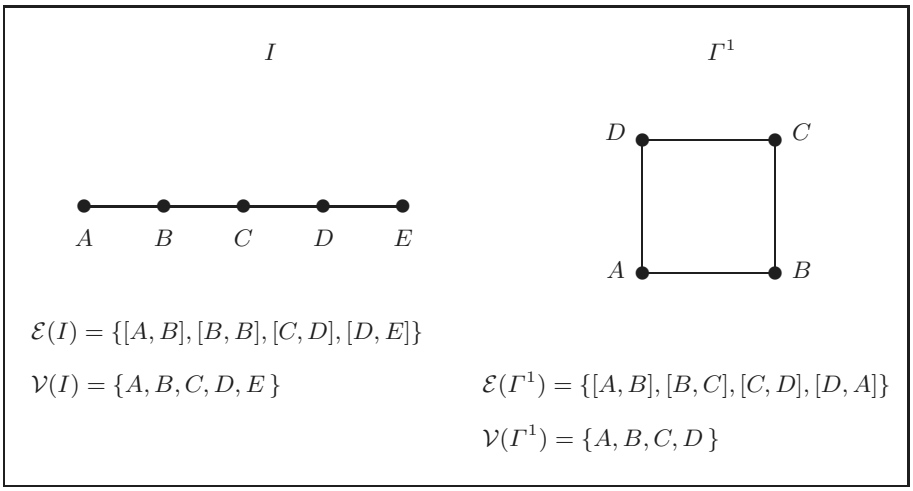


Fig. 1.11. Graphs and corresponding combinatorial graphs for $[0,1]$ and Γ^1.

right-hand column is derived from the combinatorial graph. To explain its meaning first recall that our goal is to produce an algebraic tool for understanding graphs. Instead of starting with formal definitions we are going to look for patterns. So for the moment the elements of the right-hand column can be considered to be algebraic quantities that correspond to elements of the combinatorial graph.

Table 1.1. Topological and Algebraic Boundaries in $[0,1]$

Topology	Algebra
$\mathrm{bd}\,[A,B] = \{A\} \cup \{B\}$	$\partial\widehat{[A,B]} = \widehat{A} + \widehat{B}$
$\mathrm{bd}\,[B,C] = \{B\} \cup \{C\}$	$\partial\widehat{[B,C]} = \widehat{B} + \widehat{C}$
$\mathrm{bd}\,[C,D] = \{C\} \cup \{D\}$	$\partial\widehat{[C,D]} = \widehat{C} + \widehat{D}$
$\mathrm{bd}\,[D,E] = \{D\} \cup \{E\}$	$\partial\widehat{[D,E]} = \widehat{D} + \widehat{E}$

On the topological level such basic algebraic notions as addition and subtraction of edges and points are not obvious concepts. The point of moving to an algebraic level is to allow ourselves this luxury. To distinguish between sets and algebra, we write the algebraic objects with a hat on top and allow ourselves to formally add them. For example, the topological objects such as a point $\{A\}$ and an edge $[A,B]$ become an algebraic object $\widehat{A}$ and $\widehat{[A,B]}$. Furthermore, we allow ourselves the luxury of writing expressions like $\widehat{A} + \widehat{B}$, $\widehat{[A,B]} + \widehat{[B,C]}$, or even $\widehat{A} + \widehat{A} = 2\widehat{A}$.

How should we interpret the symbol ∂, called the *boundary operator*, which we have written in the table? We are doing algebra, so it should be some type of map that takes the algebraic object $\widehat{[A,B]}$ to the sum of $\widehat{A}$ and $\widehat{B}$. The nicest maps are *linear maps*. This would mean that

$$\begin{aligned}\partial(\widehat{[A,B]}+\widehat{[B,C]}) &= \partial(\widehat{[A,B]})+\partial(\widehat{[B,C]})\\ &\overset{(1)}{=} \widehat{A}+\widehat{B}+\widehat{B}+\widehat{C}\\ &= \widehat{A}+2\widehat{B}+\widehat{C},\end{aligned}$$

where $\overset{(1)}{=}$ follows from Table 1.1.

The counterpart of this calculation on the topology level would be

$$\operatorname{bd}([A,B]\cup[B,C]) = \operatorname{bd}[A,C] = \{A\}\cup\{C\}.$$

If we think that $+$ on the algebraic side somehow matches $\cup$ on the topology side, then this suggests that we would like

$$\partial(\widehat{[A,B]}+\widehat{[B,C]}) = \widehat{A}+\widehat{C} = \partial\widehat{[A,C]}.$$

The only way that this can happen is for $2\widehat{B}=0$. This may seem like a pretty strange relation and suggests that at this point there are three things we can do:

1. Give up;
2. start over and try to find a different definition for ∂; or
3. be stubborn and press on.

The fact that this book has been written suggests that we are not about to give up. We shall discuss option 2 in Section 1.5. For now we shall just press on and adopt the trick of counting *modulo 2*. This means that we will just check whether an element appears an odd or even number of times; if it is odd we keep the element, if it is even we discard the element, that is, we declare

$$0 = 2\widehat{A} = 2\widehat{B} = 2\widehat{C} = 2\widehat{D} = 2\widehat{E}.$$

Continuing to use the presumed linearity of ∂ and counting modulo 2, we have that

$$\begin{aligned}\partial\left(\widehat{[A,B]}+\widehat{[B,C]}+\widehat{[C,D]}+\widehat{[D,E]}\right) &= \widehat{A}+\widehat{B}+\widehat{B}+\widehat{C}+\widehat{C}+\widehat{D}+\widehat{D}+\widehat{E}\\ &= \widehat{A}+\widehat{E}.\end{aligned}$$

As an indication that we are not too far off track, observe that if we had begun with a representation of I in terms of the combinatorial graph

$$\mathcal{E}'(I) = \{[A,E]\} \quad \mathcal{V}'(I) = \{A,E\},$$

then $\operatorname{bd}[A,E] = \{A\}\cup\{E\}$.

Doing the same for the graph and combinatorial graph representing Γ^1 we get Table 1.2. Adding up the algebraic boundaries, we have

$$\begin{aligned}\partial\left(\widehat{[A,B]}+\widehat{[B,C]}+\widehat{[C,D]}+\widehat{[D,A]}\right) &= \widehat{A}+\widehat{B}+\widehat{B}+\widehat{C}+\widehat{C}+\widehat{D}+\widehat{D}+\widehat{A}\\ &= \widehat{A}+\widehat{A}\\ &= 0.\end{aligned} \tag{1.4}$$

Table 1.2. Topology and Algebra of Boundaries in Γ^1

Topology	Algebra
$\mathrm{bd}\,[A,B]=\{A\}\cup\{B\}$	$\partial\widehat{[A,B]}=\widehat{A}+\widehat{B}$
$\mathrm{bd}\,[B,C]=\{B\}\cup\{C\}$	$\partial\widehat{[B,C]}=\widehat{B}+\widehat{C}$
$\mathrm{bd}\,[C,D]=\{C\}\cup\{D\}$	$\partial\widehat{[C,D]}=\widehat{C}+\widehat{D}$
$\mathrm{bd}\,[D,A]=\{D\}\cup\{A\}$	$\partial\widehat{[D,A]}=\widehat{D}+\widehat{A}$

Based on these two examples one might make the extravagant claim that spaces with *cycles*—algebraic objects whose boundaries add up to zero—enclose regions. This is almost true.

To see how this fails, observe that we could "fill in" Γ^1 [think of Figure 1.2(d)]. How does this affect the algebra? To make sense of this we need to go beyond graphs into *cubical complexes*, which will be defined later. For the moment consider the picture and collection of sets in Figure 1.12. The new aspect is the square Q that has filled in the region bounded by Γ^1. This is coded in the combinatorial information as the element $\{Q\}$.

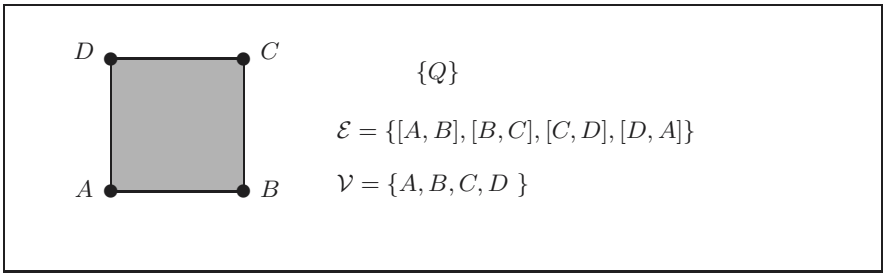


Fig. 1.12. The set Q and corresponding combinatorial data.

Observe that the perimeter or boundary of Q is Γ^1. Table 1.3 contains the topological boundary information and the associated algebra.

Table 1.3. Topology and Algebra of Boundaries in Q

Topology	Algebra
$\mathrm{bd}\, Q = [A,B] \cup [B,C] \cup [C,D] \cup [D,A]$	$\partial\widehat{Q} = \widehat{[A,B]} + \widehat{[B,C]} + \widehat{[C,D]} + \widehat{[D,A]}$
$\mathrm{bd}\,[A,B] = \{A\} \cup \{B\}$	$\partial\widehat{[A,B]} = \widehat{A} + \widehat{B}$
$\mathrm{bd}\,[B,C] = \{B\} \cup \{C\}$	$\partial\widehat{[B,C]} = \widehat{B} + \widehat{C}$
$\mathrm{bd}\,[C,D] = \{C\} \cup \{D\}$	$\partial\widehat{[C,D]} = \widehat{C} + \widehat{D}$
$\mathrm{bd}\,[D,A] = \{D\} \cup \{A\}$	$\partial\widehat{[D,A]} = \widehat{D} + \widehat{A}$

Since $\Gamma^1 \subset Q$, it is not surprising to see the contents of Table 1.2 contained in Table 1.3. Now observe that

$$\partial\widehat{Q} = \widehat{[A,B]} + \widehat{[B,C]} + \widehat{[C,D]} + \widehat{[D,A]}.$$

Equation (1.4) indicates that the cycle $\widehat{[A,B]} + \widehat{[B,C]} + \widehat{[C,D]} + \widehat{[D,A]}$ was the interesting algebraic aspect of Γ^1. In Q it appears as the boundary of an object. Our observation is that *cycles that are boundaries are uninteresting and should be ignored*.

Restating this purely algebraically, we are looking for cycles, that is elements of the *kernel* of the boundary operator. Furthermore, if a cycle is a *boundary*, namely it belongs to the *image* of the boundary operator, then we wish to ignore it. From an algebraic point of view this means we want, somehow, to set boundaries equal to zero.

The reader may have wondered why, after introducing the notion of a loop, we suddenly switched to the language of cycles. Loops are combinatorial objects—lists that our computer can store. Cycles, on the other hand, are algebraic objects. The comments of the previous paragraph have no simple conceptual correspondence on the combinatorial level.

We have by now introduced many vague and complicated notions. If you feel things are spinning out of control, don't worry. Admittedly, there are a lot of loose ends that we need to tie up, and we will begin to do so in the next chapter. The process of developing new mathematics typically involves developing new intuitions and finding new patterns—in this case we have the advantage of knowing that it will all work out in the end. For now let's just enjoy trying to match topology and algebra.

Exercises

1.4 Repeat the discussion of this section using Figures 1.2(c) and (d). In particular, make up a topology and algebra table and identify the cycles.

1.5 Repeat the above computations for a graph that represents a triangle in the plane.

1.5 Keeping Track of Directions

We want to repeat the discussion of the last section, but this time we will try to avoid counting the algebraic objects modulo 2. To do this we will consider I and I^1 as the explicit subsets of $\mathbf{R}^2$ indicated in Figure 1.13. Of course, this figure looks a lot like Figure 1.11. However, we have added arrows, which we can think of as the standard directions of the x- and y-axes.

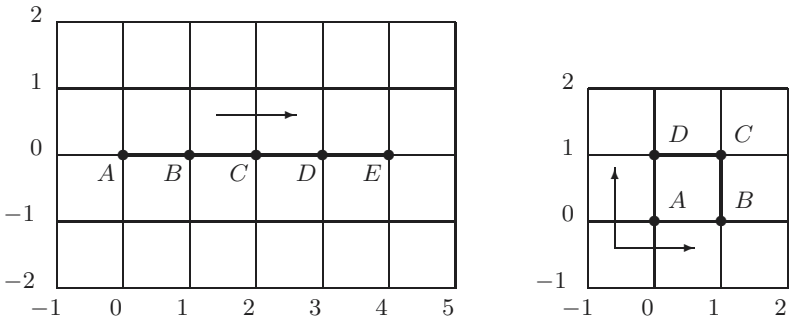


Fig. 1.13. The sets I and I^1 as explicit subsets of $\mathbf{R}^2$.

We will use these arrows to give a sense of direction to the edges and use the corresponding algebra to keep track of this as follows. Consider the edge $[A, B]$ in either I or I^1. On the combinatorial level it is defined by its endpoints A and B. Thus, in principle, we could denote the edge by $[A, B]$ or $[B, A]$. For the computations performed in the previous subsection, this would make no difference (check it). Now, however, we want to distinguish these two cases. In particular, since the x-coordinate value of the vertex A is less than that of B, we insist on writing the edge as $[A, B]$. Consider the edge with vertices C and D in I and I^1. In the first case we write $[C, D]$, but in the latter case we write $[D, C]$. In a similar vein, the edge with vertices A and D in I^1 is written as $[A, D]$ since the y-coordinate of A is less than the y-coordinate of D.

Having insisted on a specific order to denote the edges, we should not lose track of this when we apply the boundary operator. Let us declare that

$$\partial[\widehat{A, B}] := \widehat{B} - \widehat{A}$$

as a way to keep track of the fact that vertex A comes before vertex B. Of course, we will still insist that ∂ be linear.

Using this linearity on the algebra generated by the edges of I,we obtain

$$\partial\left([\widehat{A,B}]+[\widehat{B,C}]+[\widehat{C,D}]+[\widehat{D,E}]\right) = \widehat{B}-\widehat{A}+\widehat{C}-\widehat{B}+\widehat{D}-\widehat{C}+\widehat{E}-\widehat{D}$$
$$= \widehat{E}-\widehat{A}.$$

Again, we see that there is consistency between the algebra and the topology, since if we think of I as a single edge, then $\operatorname{bd} I = \{E\} \cup \{A\}$ and the minus sign suggests traversing from A to E.

Applying the boundary operator to the algebraic objects generated by the edges of Γ^1 gives rise to Table 1.4.

Table 1.4. Topology and Algebra of Boundaries in Γ^1 Remembering the Directions of the x- and y-axes.

Topology	Algebra
$\operatorname{bd}[A,B] = \{A\} \cup \{B\}$	$\partial[\widehat{A,B}] = \widehat{B} - \widehat{A}$
$\operatorname{bd}[B,C] = \{B\} \cup \{C\}$	$\partial[\widehat{B,C}] = \widehat{C} - \widehat{B}$
$\operatorname{bd}[D,C] = \{C\} \cup \{D\}$	$\partial[\widehat{D,C}] = \widehat{C} - \widehat{D}$
$\operatorname{bd}[A,D] = \{D\} \cup \{A\}$	$\partial[\widehat{A,D}] = \widehat{D} - \widehat{A}$

When we go around Γ^1 counterclockwise and keep track of the directions through which we pass the edges, we see that $[D,C]$ and $[A,D]$ are traversed in the opposite order from the orientations of the x- and y-axes. To keep track of this on the algebraic level we will write

$$[\widehat{A,B}]+[\widehat{B,C}]-[\widehat{D,C}]-[\widehat{A,D}].$$

Notice that we see a significant difference with the purely topological representation

$$\Gamma^1 = [A,B] \cup [B,C] \cup [D,C] \cup [A,D].$$

However,

$$\begin{aligned}\partial\left([\widehat{A,B}]+[\widehat{B,C}]-[\widehat{D,C}]-[\widehat{A,D}]\right) &= \partial\left([\widehat{A,B}]\right)+\partial\left([\widehat{B,C}]\right)\\ &\quad -\partial\left([\widehat{D,C}]\right)-\partial\left([\widehat{A,D}]\right)\\ &= \widehat{B}-\widehat{A}+\widehat{C}-\widehat{B}+\widehat{D}-\widehat{C}+\widehat{A}-\widehat{D}\\ &= 0.\end{aligned}$$

So again, we see that the algebra that corresponds to the interesting topology is a cycle—a sum of algebraic objects whose boundaries add up to zero.

We still need to understand what happens to this algebra when we fill in Γ^1 by Q. Consider Figure 1.14. Table 1.5 contains the topological boundary information and algebra that we are associating to it.

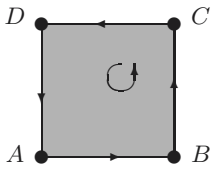


Fig. 1.14. The set Q and associated directions.

Since $\Gamma^1 \subset Q$, we again see the contents of Table 1.4 contained in Table 1.5. Keeping track of directions and walking around the edge of Q in a counterclockwise direction, it seems reasonable to define

$$\partial\widehat{Q} = [\widehat{A,B}] + [\widehat{B,C}] - [\widehat{D,C}] - [\widehat{A,D}].$$

Equation (1.4) indicates that the cycle $[\widehat{A,B}] + [\widehat{B,C}] - [\widehat{D,C}] - [\widehat{A,D}]$ is the interesting algebraic aspect of Γ^1. In Q it appears as the boundary of an object. Again, the observation that we will make is: Cycles that are boundaries should be considered uninteresting.

Table 1.5. Topology and Algebra of Boundaries in Figure 1.14

Topology	Algebra
$\mathrm{bd}\, Q = \Gamma^1 = [A,B] \cup [B,C] \cup [C,D] \cup [D,A]$	$\partial\widehat{Q} = [\widehat{A,B}] + [\widehat{B,C}] - [\widehat{D,C}] - [\widehat{A,D}]$
$\mathrm{bd}\,[A,B] = \{A\} \cup \{B\}$	$\partial[\widehat{A,B}] = \widehat{B} - \widehat{A}$
$\mathrm{bd}\,[B,C] = \{B\} \cup \{C\}$	$\partial[\widehat{B,C}] = \widehat{C} - \widehat{B}$
$\mathrm{bd}\,[D,C] = \{D\} \cup \{C\}$	$\partial[\widehat{D,C}] = \widehat{C} - \widehat{D}$
$\mathrm{bd}\,[A,D] = \{A\} \cup \{D\}$	$\partial[\widehat{A,D}] = \widehat{D} - \widehat{A}$

1.6 Mod 2 Homology of Graphs

We have done the same example twice using different types of arithmetic, but the conclusion is the same. We should look for a linear operator that somehow algebraically mimics what is done by taking the edge or boundary of a set. Then, having found this operator, we should look for cycles (elements of the kernel) but ignore boundaries (elements of the image). This is still pretty fuzzy so let's do it again; a little slower and more formally, but in the general

setting of graphs and using linear algebra. We are not yet aiming for rigor, but we do want to suggest that there is a way to deal with all the ideas that are being thrown about in a systematic manner.

The linear algebra we are going to do, *mod 2 linear algebra*, may seem a little strange at first, but hopefully you will recognize that all the important ideas, such as vector addition, basis, matrices, etc., still hold. The difference is that unlike the traditional linear algebra, where the set of scalars by which vectors may be multiplied consists of all real numbers, in the mod 2 linear algebra we restrict the scalars to the set $\mathbf{Z}_2 := \{0, 1\}$. A nice consequence is that the number of algebraic elements is finite, so in many cases we can explicitly write them out.

Let G be a graph with a fixed representation as a combinatorial graph with vertices $\mathcal{V}$ and edges $\mathcal{E}$. We will construct two vector spaces $C_0(G; \mathbf{Z}_2)$ and $C_1(G; \mathbf{Z}_2)$ as follows. Declare the set of vertices $\mathcal{V}$ to be the set of basis elements of $C_0(G; \mathbf{Z}_2)$. Thus, if $\mathcal{V} = \{v_1, \ldots, v_n\}$, then the collection

$$\{\widehat{v}_i \mid i = 1, \ldots, n\}$$

is a basis for $C_0(G; \mathbf{Z}_2)$. Notice that we are continuing to use the hat notation to distinguish between an algebraic object (a basis element) and a combinatorial object (an element in a list).[2] Since $\{\widehat{v}_i \mid i = 1, \ldots, n\}$ is a basis, elements of $C_0(G; \mathbf{Z}_2)$ take the form

$$c = \alpha_1 \widehat{v}_1 + \alpha_2 \widehat{v}_2 + \cdots + \alpha_n \widehat{v}_n, \tag{1.5}$$

where $\alpha_i \in \mathbf{Z}_2$. An expression of this form will be referred to as a *chain*. Note that the symbol $\mathbf{Z}_2$ in the notation $C_0(G; \mathbf{Z}_2)$ and $C_1(G; \mathbf{Z}_2)$ is used to remind us that we are adding mod 2.

Example 1.4 Let us assume that $\mathcal{V} = \{v_1, v_2, v_3, v_4\}$. Then the basis elements for $C_0(G; \mathbf{Z}_2)$ are $\{\widehat{v_1}, \widehat{v_2}, \widehat{v_3}, \widehat{v_4}\}$. Therefore, every element of $C_0(G; \mathbf{Z}_2)$ can be written as

$$c = \alpha_1 \widehat{v_1} + \alpha_2 \widehat{v_2} + \alpha_3 \widehat{v_3} + \alpha_4 \widehat{v_4}.$$

However, since $\alpha_i \in \{0, 1\}$, we can write out all the elements of $C_0(G; \mathbf{Z}_2)$. In particular,

[2] Since we are trying to be a little more formal in this section, perhaps this is a good place to emphasize the fact that elements of $\mathcal{V}$ are really combinatorial objects. These are the objects that we want the computer to manipulate; therefore, they are just elements of a list stored in the computer. Of course, since we are really interested in understanding the topology of the graph G, we make the identification of $v \in \mathcal{V}$ with $v \in G \subset \mathbf{R}^3$. But it is we who make this identification, not the computer. Furthermore, as will become clear especially in Part II where we discuss the applications of homology, the ability to make this identification, and thereby pass from combinatorics to topology, is a very powerful technique.

$$C_0(G;\mathbf{Z}_2) = \left\{ \begin{array}{c} 0,\ \widehat{v_1},\ \widehat{v_2},\ \widehat{v_3},\ \widehat{v_4}, \\ \widehat{v_1}+\widehat{v_2},\ \widehat{v_1}+\widehat{v_3},\ \widehat{v_1}+\widehat{v_4},\ \widehat{v_2}+\widehat{v_3},\ \widehat{v_2}+\widehat{v_4},\ \widehat{v_3}+\widehat{v_4}, \\ \widehat{v_1}+\widehat{v_2}+\widehat{v_3},\ \widehat{v_1}+\widehat{v_2}+\widehat{v_4},\ \widehat{v_1}+\widehat{v_3}+\widehat{v_4},\ \widehat{v_2}+\widehat{v_3}+\widehat{v_4}, \\ \widehat{v_1}+\widehat{v_2}+\widehat{v_3}+\widehat{v_4} \end{array} \right\},$$

Each element of $C_0(G;\mathbf{Z}_2)$ is a vector and, of course, we are allowed to add vectors, but mod 2, for example,

$$(\widehat{v_1}+\widehat{v_2}+\widehat{v_3})+(\widehat{v_1}+\widehat{v_2}+\widehat{v_4}) = 2\widehat{v_1}+2\widehat{v_2}+\widehat{v_3}+\widehat{v_4} = 0+0+\widehat{v_3}+\widehat{v_4} = \widehat{v_3}+\widehat{v_4}.$$

Returning to the general discussion, let the set of edges $\mathcal{E}$ be the set of basis elements of $C_1(G;\mathbf{Z}_2)$. If $\mathcal{E} = \{e_1,\ldots,e_k\}$, then the collection $\{\widehat{e_i} \mid i = 1,\ldots,k\}$ is a basis for $C_1(G;\mathbf{Z}_2)$ and an element of $C_1(G;\mathbf{Z}_2)$ takes the form

$$c = \alpha_1\widehat{e}_1 + \alpha_2\widehat{e}_2 + \cdots + \alpha_k\widehat{e}_k,$$

where again α_i is 0 or 1. The vector spaces $C_k(G;\mathbf{Z}_2)$ are called the *k-chains* for G. A word of warning is in order here. The vector space is an algebraic concept based on very geometric ideas. However, the reader should not seek any relation between the geometry of the vector spaces $C_k(G;\mathbf{Z}_2)$ and the geometry of the graph G. As we will see soon, even the dimensions of $C_k(G;\mathbf{Z}_2)$ have nothing to do with the dimension 3 of the space in which G is located.

It is convenient to introduce two more vector spaces $C_2(G;\mathbf{Z}_2)$ and $C_{-1}(G;\mathbf{Z}_2)$. We will always take $C_{-1}(G;\mathbf{Z}_2)$ to be the trivial vector space $\mathbf{0}$, that is, the vector space consisting of exactly one element 0. For graphs we will also set $C_2(G;\mathbf{Z}_2)$ to be the trivial vector space. As we will see in Chapter 2, for higher-dimensional spaces this is not the case.

We now need to formally define the boundary operators that were alluded to earlier. Let

$$\begin{aligned} \partial_0 &: C_0(G;\mathbf{Z}_2) \to C_{-1}(G;\mathbf{Z}_2), \\ \partial_1 &: C_1(G;\mathbf{Z}_2) \to C_0(G;\mathbf{Z}_2), \\ \partial_2 &: C_2(G;\mathbf{Z}_2) \to C_1(G;\mathbf{Z}_2) \end{aligned}$$

be *linear maps*. Since $C_{-1}(G;\mathbf{Z}_2) = \mathbf{0}$, it is clear that the image of ∂_0 must be zero. For similar reasons, the same must be true for ∂_2. Since we have chosen bases for the vector spaces $C_1(G;\mathbf{Z}_2)$ and $C_0(G;\mathbf{Z}_2)$, we can express ∂_1 as a matrix. The entries of this matrix are determined by how ∂_1 acts on the basis elements (i.e., the edges $\mathbf{e}_i$). In line with the previous discussion we make the following definition. Let the edge e_i have vertices v_j and v_k. Define

$$\partial_1\widehat{e}_i := \widehat{v}_j + \widehat{v}_k.$$

In our earlier example we were interested in cycles, that is, objects that get mapped to 0 by ∂. In general, given a linear map A, the set of elements that get sent to 0 is called the *kernel* of A and is denoted by $\ker A$. Thus the

set of cycles forms the kernel of ∂. Because the set of cycles plays such an important role, it has its own notation:

$$Z_0(G;\mathbf{Z}_2) := \ker \partial_0 = \{c \in C_0(G;\mathbf{Z}_2) \mid \partial_0 c = 0\},$$
$$Z_1(G;\mathbf{Z}_2) := \ker \partial_1 = \{c \in C_1(G;\mathbf{Z}_2) \mid \partial_1 c = 0\}.$$

Since $C_{-1}(G;\mathbf{Z}_2) = 0$, it is obvious that $Z_0(G;\mathbf{Z}_2) = C_0(G;\mathbf{Z}_2)$; namely everything in $C_0(G;\mathbf{Z}_2)$ gets sent to 0.

We also observed that cycles that are boundaries are not interesting. To formally state this, define the set of boundaries to be the image of the boundary operator. More precisely,

$$B_0(G;\mathbf{Z}_2) := \operatorname{im} \partial_1 = \{b \in C_0(G;\mathbf{Z}_2) \mid \exists\, c \in C_1(G;\mathbf{Z}_2) \text{ such that } \partial_1 c = b\},$$
$$B_1(G;\mathbf{Z}_2) := \operatorname{im} \partial_2 = \{b \in C_1(G;\mathbf{Z}_2) \mid \exists\, c \in C_2(G;\mathbf{Z}_2) \text{ such that } \partial_2 c = b\}.$$

Recall that we set $C_2(G;\mathbf{Z}_2) = \mathbf{0}$; thus $B_1(G;\mathbf{Z}_2) = \mathbf{0}$.

Observe that $B_0(G;\mathbf{Z}_2) \subset C_0(G;\mathbf{Z}_2) = Z_0(G;\mathbf{Z}_2)$, that is, every 0-boundary is a 0-cycle. We shall show later that every boundary is a cycle, that is,

$$B_k(G;\mathbf{Z}_2) \subset Z_k(G;\mathbf{Z}_2).$$

This is a very important fact—but not at all obvious at this point.

We can finally define homology in this rather special setting. For $k = 0, 1$, the kth homology with $\mathbf{Z}_2$ coefficients is defined to be the quotient space

$$H_k(G;\mathbf{Z}_2) := Z_k(G;\mathbf{Z}_2)/B_k(G;\mathbf{Z}_2).$$

If you have not worked with quotient spaces, then the obvious question is: *What does this notation mean?*

Let us step back for a moment and remember what we are trying to do. Recall that the interesting objects are cycles, but if a cycle is a boundary, then it is no longer interesting. Thus we begin with a cycle $z \in Z_k(G;\mathbf{Z}_2)$. It is possible that $z \in B_k(G;\mathbf{Z}_2)$. In this case we want z to be uninteresting. From an algebraic point of view we can take this to mean that we want to set z equal to 0.

Now consider two cycles $z_1, z_2 \in Z_i(G;\mathbf{Z}_2)$. What if there exists a boundary $b \in B_k(G;\mathbf{Z}_2)$ such that

$$z_1 + b = z_2?$$

Since boundaries are supposed to be 0, this suggests that b should be zero and hence that we want z_1 and z_2 to be the same.

Mathematically, when we want different objects to be the same, we form *equivalence classes*. With this in mind we define an equivalence class on the set of cycles by

$$z_1 \sim z_2 \quad \text{if and only if} \quad z_1 + b = z_2$$

for some $b \in B_k(G; \mathbf{Z}_2)$. The notation $\sim$ means equivalent. The equivalence class of the cycle $z \in Z_k(G; \mathbf{Z}_2)$ is the set of all cycles that are equivalent to z. It is denoted by $[z]$. Thus

$$[z] := \{z' \in Z_k(G; \mathbf{Z}_2) \mid z \sim z'\}.$$

Therefore, $z_1 \sim z_2$ is the same as saying $[z_1] = [z_2]$. The set of equivalence classes makes up the elements of $H_k(G; \mathbf{Z}_2)$.

This brief discussion contains several fundamental but nontrivial mathematical concepts, so to help make some of these ideas clearer consider the following examples.

Example 1.5 Let us start with the trivial graph consisting of a single point, $G = \{v\}$. Then

$$\mathcal{V} = \{v\}, \quad \mathcal{E} = \emptyset.$$

These are used to generate the bases for the chains. In particular, the basis for $C_0(G; \mathbf{Z}_2)$ is $\{\widehat{v}\}$. This means that any element $v \in C_0(G; \mathbf{Z}_2)$ has the form

$$c = \alpha\widehat{v}.$$

If $\alpha = 0$, then $c = 0$. If $\alpha = 1$, then $c = \widehat{v}$. Thus

$$C_0(G; \mathbf{Z}_2) = \{0, \widehat{v}\}.$$

Since, by definition, $\partial_0 = 0$, it follows that $Z_0(G; \mathbf{Z}_2) = C_0(G; \mathbf{Z}_2)$. Thus

$$Z_0(G; \mathbf{Z}_2) = \{0, \widehat{v}\}.$$

The basis for $C_1(G; \mathbf{Z}_2)$ is the empty set; therefore,

$$C_1(G; \mathbf{Z}_2) = \mathbf{0}.$$

By definition, $Z_1(G; \mathbf{Z}_2) \subset C_1(G; \mathbf{Z}_2)$, so $Z_1(G; \mathbf{Z}_2) = \mathbf{0}$.

Since $C_1(G; \mathbf{Z}_2) = \mathbf{0}$, the image of $C_1(G; \mathbf{Z}_2)$ under ∂_1 must also be trivial. Thus $B_0(G; \mathbf{Z}_2) = \mathbf{0}$. Now consider two cycles $z_1, z_2 \in Z_0(G; \mathbf{Z}_2)$. $B_0(G; \mathbf{Z}_2) = \mathbf{0}$ implies that $z_1 \sim z_2$ if and only if $z_1 = z_2$. Therefore,

$$H_0(G; \mathbf{Z}_2) = Z_0(G; \mathbf{Z}_2) = \{[0], [\widehat{v}]\}.$$

Of course, since $Z_1(G; \mathbf{Z}_2) = \mathbf{0}$, it follows that

$$H_1(G; \mathbf{Z}_2) = \mathbf{0}.$$

Example 1.6 Let G be the graph of Figure 1.11 representing I. Then,

$$\mathcal{V} = \{A, B, C, D, E\},$$
$$\mathcal{E} = \{[A, B], [B, C][C, D], [D, E]\}.$$

Thus the basis for the 0-chains, $C_0(G;\mathbf{Z}_2)$, is

$$\{\widehat{A},\widehat{B},\widehat{C},\widehat{D},\widehat{E}\},$$

while the basis for the 1-chains, $C_1(G;\mathbf{Z}_2)$, is

$$\{[\widehat{A,B}],[\widehat{B,C}],[\widehat{C,D}],[\widehat{D,E}]\}.$$

Unlike the previous example, we really need to write down the boundary operator $\partial_1 : C_1(G;\mathbf{Z}_2) \to C_0(G;\mathbf{Z}_2)$. Given the above-mentioned bases, since ∂_1 is a linear map it can be written as a matrix. To do this it is convenient to use the notation of column vectors. So let

$$\widehat{A} = \begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix},\ \widehat{B} = \begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix},\ \widehat{C} = \begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix},\ \widehat{D} = \begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix},\ \widehat{E} = \begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}$$

and

$$[\widehat{A,B}] = \begin{bmatrix}1\\0\\0\\0\end{bmatrix},\ [\widehat{B,C}] = \begin{bmatrix}0\\1\\0\\0\end{bmatrix},\ [\widehat{C,D}] = \begin{bmatrix}0\\0\\1\\0\end{bmatrix},\ [\widehat{D,E}] = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

With this convention, ∂_1 becomes the 5×4 matrix

$$\partial_1 = \begin{bmatrix}1&0&0&0\\1&1&0&0\\0&1&1&0\\0&0&1&1\\0&0&0&1\end{bmatrix}.$$

Let's do a quick check. For example,

$$\partial_1[\widehat{B,C}] = \begin{bmatrix}1&0&0&0\\1&1&0&0\\0&1&1&0\\0&0&1&1\\0&0&0&1\end{bmatrix}\begin{bmatrix}0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}0\\1\\1\\0\\0\end{bmatrix} = \widehat{B}+\widehat{C}.$$

The next step is to compute the cycles—to find $Z_1(G;\mathbf{Z}_2) := \ker \partial_1$. Observe that by definition the chain $c \in C_1(G;\mathbf{Z}_2)$ is in $Z_1(G;\mathbf{Z}_2)$ if and only if $\partial_1 c = \widehat{0}$. Again, since $c \in C_1(G;\mathbf{Z}_2)$, it can be written as a sum of basis elements, that is,

$$c = \alpha_1[\widehat{A,B}] + \alpha_2[\widehat{B,C}] + \alpha_3[\widehat{C,D}] + \alpha_4[\widehat{D,E}].$$

Writing this in the form of a column vector, we have

$$c = \alpha_1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \alpha_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix}.$$

In this form, finding $c \in \ker \partial_1$ is equivalent to solving the equation

$$\partial_1 \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 0\,1\,1\,0 \\ 0\,0\,1\,1 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} \alpha_1 \\ \alpha_1 + \alpha_2 \\ \alpha_2 + \alpha_3 \\ \alpha_3 + \alpha_4 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix},$$

which implies that $\alpha_i = 0$ for $i = 1, \ldots, 4$. Thus the only element in $Z_1(G; \mathbf{Z}_2)$ is the zero vector and hence $Z_1(G; \mathbf{Z}_2) = \mathbf{0}$. Since $\partial_2 = 0$, we have $B_1(G; \mathbf{Z}_2) = \mathbf{0}$. So

$$H_1(G; \mathbf{Z}_2) := Z_1(G; \mathbf{Z}_2)/B_1(G; \mathbf{Z}_2) = \mathbf{0}.$$

Computing $H_0(G; \mathbf{Z}_2)$ is more interesting. Since $Z_0(G; \mathbf{Z}_2) = C_0(G; \mathbf{Z}_2)$, a basis for $Z_0(G; \mathbf{Z}_2)$ consists of

$$\widehat{A} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \widehat{B} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ \widehat{C} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \ \widehat{D} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \ \widehat{E} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

What about $B_0(G; \mathbf{Z}_2)$? This is the image of ∂_1 spanned by the images of each of the basis elements. This we can compute. In particular,

$$\partial_1[\widehat{A, B}] = \begin{bmatrix} 1\,0\,0\,0 \\ 1\,1\,0\,0 \\ 0\,1\,1\,0 \\ 0\,0\,1\,1 \\ 0\,0\,0\,1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Similarly,

$$\partial_1[\widehat{B, C}] = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \partial_1[\widehat{C, D}] = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad \partial_1[\widehat{D, E}] = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

From this we can conclude that if $b \in B_0(G; \mathbf{Z}_2)$, then

$$b = \alpha_1 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \alpha_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix},$$

where $\alpha_i \in \{0, 1\}$.

It is easy to check that no basis element of $Z_0(G; \mathbf{Z}_2)$ is in $B_0(G; \mathbf{Z}_2)$. For example, if for some $\alpha_1, \alpha_2, \alpha_3, , \alpha_4, \alpha_5 \in \{0, 1\}$ we have

$$\widehat{A} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \alpha_1 \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \alpha_2 \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \alpha_3 \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} + \alpha_4 \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix},$$

then

$$\begin{aligned} \alpha_1 &= 1, \\ \alpha_1 + \alpha_2 &= 0, \\ \alpha_2 + \alpha_3 &= 0, \\ \alpha_3 + \alpha_4 &= 0, \\ \alpha_4 &= 0. \end{aligned}$$

Since we count modulo 2, we conclude from the second equation that $\alpha_2 = 1$. But then similarly also $\alpha_3 = \alpha_4 = 1$, which contradicts the last equation. This leads to the conclusion $[\widehat{A}] \neq [0]$ for the equivalence classes in $H_0(G; \mathbf{Z}_2)$. On the other hand, again counting mod 2,

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We can rewrite this equation as

$$\widehat{A} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \widehat{B}.$$

Since

$$\begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in B_0(G; \mathbf{Z}_2),$$

this implies that

$$\widehat{A} \sim \widehat{B}.$$

Therefore, on the level of homology

$$[\widehat{A}] = [\widehat{B}] \in H_0(G; \mathbf{Z}_2).$$

We leave it to the reader to check that, in fact,

$$[\widehat{A}] = [\widehat{B}] = [\widehat{C}] = [\widehat{D}] = [\widehat{E}] \in H_0(G; \mathbf{Z}_2). \tag{1.6}$$

Thus, if we write out the elements of $H_0(G; \mathbf{Z}_2)$ without repetition, we get

$$\{[0], [\widehat{A}]\} \subset H_0(G; \mathbf{Z}_2). \tag{1.7}$$

The question is if we have found all homology classes in $H_0(G; \mathbf{Z}_2)$. To answer it we need to understand that the vector space $Z_0(G; \mathbf{Z}_2)$ *induces* the structure of a vector space on $H_0(G; \mathbf{Z}_2)$. For this end consider the two cycles $\widehat{A}, \widehat{B}$. They both induce the same element in $H_0(G; \mathbf{Z}_2)$ because $[\widehat{A}] = [\widehat{B}]$. Since $Z_0(G; \mathbf{Z}_2)$ is a vector space, we can add $\widehat{A}$ and $\widehat{B}$ to get $\widehat{A} + \widehat{B}$. Observe that in column notation

$$\widehat{A} + \widehat{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \in B_0(G; \mathbf{Z}_2).$$

Let us collect all this information in the context of the equivalence classes.

$$[0] \overset{1}{=} 2[\widehat{A}] = [\widehat{A}] + [\widehat{A}] \overset{2}{=} [\widehat{A}] + [\widehat{B}] \overset{3}{=} [\widehat{A} + \widehat{B}] \overset{4}{=} [0].$$

Equality 1 follows from the fact that we are still doing mod 2 arithmetic. Equality 2 holds because $\widehat{A} \sim \widehat{B}$. Equality 3 is what we mean by saying that $Z_0(G; \mathbf{Z}_2)$ induces the structure of a vector space on $H_0(G; \mathbf{Z}_2)$. More precisely, to add equivalence classes we just add representatives of the equivalence classes.[3] The last equality follows from the fact that $\widehat{A} + \widehat{B} \in B_0(G; \mathbf{Z}_2)$.

Equipped with the vector space structure of $H_0(G; \mathbf{Z}_2)$, we easily see that the homology class of an arbitrary cycle is

$$[\alpha_1\widehat{A} + \alpha_2\widehat{B} + \alpha_3\widehat{C} + \alpha_4\widehat{D} + \alpha_5\widehat{E}] = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4 + \alpha_5)[\widehat{A}] \in \{[0], [\widehat{A}]\}.$$

Therefore,

$$H_0(G; \mathbf{Z}_2) = \{[0], [\widehat{A}]\}. \tag{1.8}$$

[3] Of course, it needs to be checked that this addition is well defined. This is done in Chapter 13.

A final comment is needed before ending this example. We motivated computing homology by arguing that we wanted an algebraic method for determining whether pictures have bounded regions or not. We introduced the combinatorial graphs as a combinatorial representation that was then turned into algebra. We do not want the final answer to depend on the graph. We also mentioned that proving this is difficult. But to emphasize this we will write the homology in terms of the set rather than the graph. In other words, what we claim (without proof!) is that

$$H_1(I;\mathbf{Z}_2) = \mathbf{0} \quad \text{and} \quad H_0(I;\mathbf{Z}_2) = \{[0],[\widehat{A}]\}.$$

Notice, however, that the chains depend explicitly on the combinatorial graph; therefore, it makes no sense to write $C_k(I;\mathbf{Z}_2)$.

Example 1.7 Let G be the graph of Figure 1.11 representing Γ^1. Then,

$$\begin{aligned}\mathcal{V} &= \{A,B,C,D\},\\ \mathcal{E} &= \{[A,B],[B,C],[D,C],[A,D]\}.\end{aligned}$$

The computation of the homology begins just as in the previous example. The basis for the 0-chains, $C_0(G;\mathbf{Z}_2)$, is

$$\{\widehat{A},\widehat{B},\widehat{C},\widehat{D}\},$$

while the basis for the 1-chains, $C_1(G;\mathbf{Z}_2)$, is

$$\{[\widehat{A,B}],[\widehat{B,C}],[\widehat{D,C}],[\widehat{A,D}]\}.$$

Again, using column vectors let

$$\widehat{A} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix},\ \widehat{B} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix},\ \widehat{C} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix},\ \widehat{D} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$$

and

$$[\widehat{A,B}] = \begin{bmatrix}1\\0\\0\\0\end{bmatrix},\ [\widehat{B,C}] = \begin{bmatrix}0\\1\\0\\0\end{bmatrix},\ [\widehat{D,C}] = \begin{bmatrix}0\\0\\1\\0\end{bmatrix},\ [\widehat{A,D}] = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

With this convention, ∂_1 becomes the 4×4 matrix

$$\partial_1 = \begin{bmatrix}1&0&0&1\\1&1&0&0\\0&1&1&0\\0&0&1&1\end{bmatrix}.$$

To compute $Z_1(G;\mathbf{Z}_2) := \ker \partial_1$, we need to solve the equation

$$\partial_1 \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \end{bmatrix} = \begin{bmatrix} \alpha_1 + \alpha_4 \\ \alpha_1 + \alpha_2 \\ \alpha_2 + \alpha_3 \\ \alpha_3 + \alpha_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

Observe that, since we are using mod 2 arithmetic, $-1 = 1$ and any solution must satisfy

$$\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4.$$

In particular, $\alpha_1 = \alpha_2 = \alpha_3 = \alpha_4 = 1$ is the only nonzero solution. Thus

$$Z_1(G;\mathbf{Z}_2) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

As in the previous example, $B_1(G;\mathbf{Z}_2) = \mathbf{0}$. So

$$H_1(\Gamma^1;\mathbf{Z}_2) := Z_1(G;\mathbf{Z}_2) = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}.$$

Notice that this is different from the previous two examples, which has to do with the fact that Γ^1 encloses a region in the plane.

We still need to compute $H_0(\Gamma^1;\mathbf{Z}_2)$. We know that $Z_0(G;\mathbf{Z}_2) = C_0(G;\mathbf{Z}_2)$. The next step is to understand $B_0(G;\mathbf{Z}_2)$. Again, we look at how ∂_1 acts on the basis elements of $C_1(G;\mathbf{Z}_2)$ and conclude that

$$\partial_1[\widehat{A,B}] = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \partial_1[\widehat{B,C}] = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \quad \partial_1[\widehat{D,C}] = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \quad \partial_1[\widehat{A,D}] = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

As before, $\widehat{A} \notin B_0(G;\mathbf{Z}_2)$, hence $[\widehat{A}] \neq [0]$ for elements of $H_0(\Gamma^1;\mathbf{Z}_2)$, but

$$\widehat{A} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \widehat{B},$$

and so

$$\widehat{A} \sim \widehat{B}.$$

It is left to the reader to check that, as in the previous example,

$$H_0(\Gamma^1;\mathbf{Z}_2) = \{[0], [\widehat{A}]\}.$$

Remark 1.8 Several times in this section we have ended up with a vector space containing two elements $\{[0], [\widehat{A}]\}$. Observe that we have the following relations under vector addition with mod 2 arithmetic:

$$[0] + [0] = [0], \quad [0] + [\widehat{A}] = [\widehat{A}], \quad [\widehat{A}] + [\widehat{A}] = [0].$$

We can identify this with mod 2 arithmetic, that is, we can consider the set $\mathbf{Z}_2 = \{0, 1\}$ where

$$0 + 0 = 0, \quad 0 + 1 = 1, \quad 1 + 1 = 0.$$

From now on we will do this. In particular, this allows us to write the homology groups from the previous example as

$$H_0(\Gamma^1; \mathbf{Z}_2) \cong \mathbf{Z}_2, \quad H_1(\Gamma^1; \mathbf{Z}_2) \cong \mathbf{Z}_2.$$

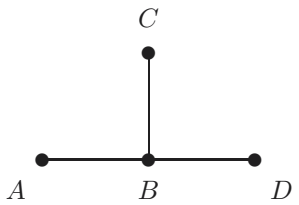
Exercises

1.6 Verify Eq. (1.6).

1.7 Verify Eq. (1.8) by explicitly writing out all the elements of $Z_0(G; \mathbf{Z}_2)$ and for each of them find out whether its homology class is 0 or $[\widehat{A}]$.

1.8 Let G be the graph with edges $\mathcal{E} = \{[v_1, v_2]\}$ and vertices $\mathcal{V} = \{v_1, v_2\}$. Compute $H_k(G; \mathbf{Z}_2)$ for $k = 0, 1$.

1.9 Compute $H_k(G; \mathbf{Z}_2)$ for $k = 0, 1$, where G is the graph



1.10 Compute $H_k(G; \mathbf{Z}_2)$ for $k = 0, 1$, where G is the graph with vertices $\{v_1, v_2, v_3\}$ and edges $\{[v_1, v_2], [v_2, v_3], [v_3, v_1]\}$.

1.11 Let G^n be a graph with n vertices where every pair of distinct vertices is connected by one edge. Create input files for the Homology program for several values of n starting from $n = 4$. Use the Homology program to compute the dimensions r_k of $H_k(G^n; \mathbf{Z}_2)$. Make a conjecture about the formula for every $n \geq 3$.

Springer

http://www.springer.com/978-0-387-40853-8

Computational Homology

Kaczynski, T.; Mischaikow, K.; Mrozek, M.

2004, XVIII, 482 p., Hardcover
ISBN: 978-0-387-40853-8