7 Unsigned Gauss codes

In the last lecture, we saw a simple algorithm to test whether a signed Gauss code is consistent with a generic curve in the plane: Count the faces (by symbol-chasing) and return true if and only if the number of faces is exactly two more than the number of crossings.

In his unpublished notes, written around 1840, Gauss was asked how to determine whether an *unsigned* Gauss code is consistent with a planar curve [3]. The same question was published about 40 years later by Tait [8]. I regard this as the first problem in computational topology. (Euler's famous Bridges of Königsberg is the *zeroth* problem in computational topology.) B oht Gauss and Tait described a *partial* solution to his problem. The first complete solution was proposed by Julius Nagy almost a century later [4].[1]

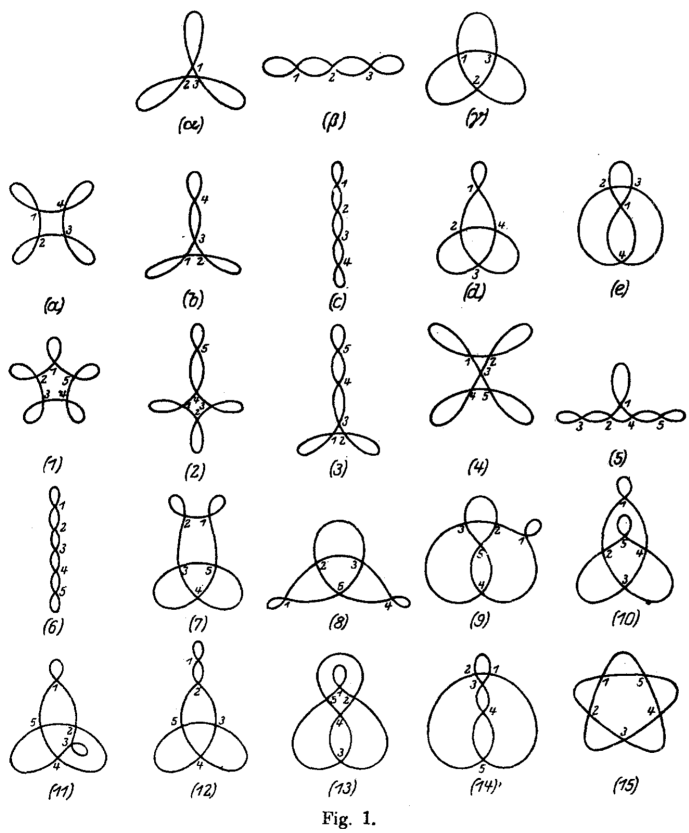

Figure 1: Nagy's 1927 census of unsigned Gauss codes of lengths 6 through 10

In this lecture, I'll describe an $O(n^2)$-time *algorithm* originally described by Max Dehn in 1936 [1], with some simplifications suggested by Nagy's solution and more modern graph algorithms, as suggested by Read and Rosenstiehl [11], Rosenstiehl and Tarjan [12], and de Fraysseix and Ossona de Mendez [3].[2] My presentation of Dehn's algorithm also closely follows Kaufmann. ***[[Fix reference numbers]]***

[1] Nagy's algorithm attempts to construct a *Seifert decomposition* of the encoded curve. I'm afraid I don't understand Nagy's solution well enough to describe it, or even to be confident that it is correct.

[2] ***Dozens*** of other combinatorial and algebraic characterizations of planar Gauss codes have been published since Dehn's solution, but as far as I know, none lead to a simpler or more efficient algorithm (except through the use of linear-time algorithms to test graph planarity, which is cheating).

7.1 Winding numbers again

Recall that the winding number of a polygon P around a point o can be computed by shooting a vertical ray from o and counting positive and negative crossings with the polygon. The same characterization extends to generic curves, but it's a little unsatisfying, for a couple of reasons. First, we only care about curves *up to isotopy*, but the ray-shooting algorithm requires choosing a specific (arbitrary) curve in the isotopy class. We can soften this objection somewhat by observing that we don't really need to count crossings with a *ray*; any path from o to infinity that crosses the curve a finite number of times will work.

But there's a more serious objection. Our representation of closed curves (signed Gauss codes) doesn't include any geometric information. But how do we represent the point? We can't give *coordinates*, because different curves with the same representation have different winding numbers around any *fixed* point!

Instead, we specify the obstacle point o by declaring which *face* of the curve contains it. More cleanly, we define the winding number of a curve γ around each of its *faces* using an *Alexander numbering*, we defined in Lecture 2 for polygons. For any directed edge e of the image graph, let $\mathsf{left}(e)$ and $\mathsf{right}(e)$ denote the faces immediately to the left and right of e (relative to the orientation of e).

- If f is the outer face, then $\mathsf{wind}(\gamma, f) = 0$.
- For every directed edge e, we have $\mathsf{wind}(\mathsf{left}(e)) = \mathsf{wind}(\mathsf{right}(e)) + 1$

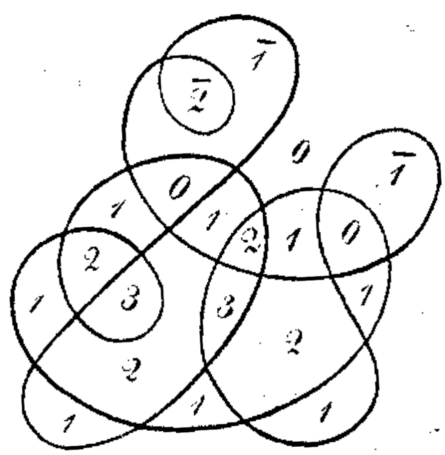

Figure 2: The Alexander numbering of a curve; bars indicate negation (Listing 1847)

7.2 Smoothing

Gauss observed that we can also define winding numbers by *smoothing* the curve at each vertex. Smoothing replaces a neighborhood of a single vertex with a pair of disjoint curve segments. In fact there are two different smoothing operations, depending on how the disjoint curve segments are attached. One smoothing operation disconnects the curve (or connects two constituents of a multicurve) but preserves the direction of both subcurves. The other keeps the curve connected, but requires the direction of part of the curve to be reversed.

Gauss observed that by smoothing *every* vertex of a curve to preserve direction, we can decompose the curve into a finite set of disjoint *simple* curves. This collection of simple curves is called the

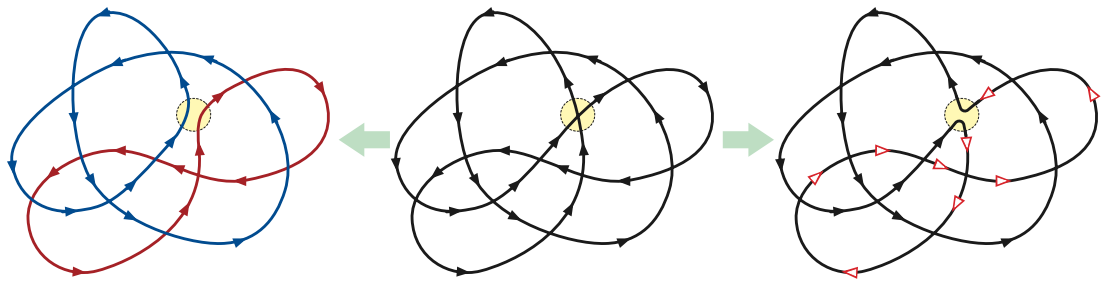

Figure 3: Smoothing a curve at a vertex. Left: preserving direction. Right: preserving connection.

Seifert decomposition of the curve. Each of these simple curves has winding number $+1$ or -1 around its interior, depending whether the curve is oriented counterclockwise or clockwise.

The winding number of a curve γ around any point o (far from the vertices) is equal to the sum of the winding numbers of the curves in the Seifert decomposition of γ around o. Equivalently, $\mathsf{wind}(\gamma, 0)$ is equal to the number of counterclockwise cycles that contain o minus the number of clockwise cycles that contain o.

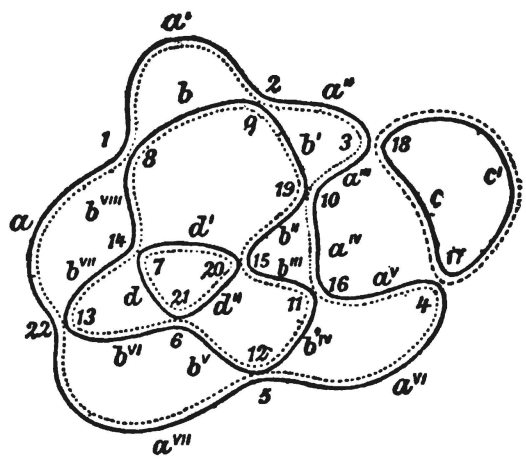

Figure 4: Gauss's example of a Seifert decomposition

7.3 Gauss's parity condition

Gauss observed without proof that the unsigned Gauss code of every planar curve satisfies a simple parity condition: Every substring that starts and ends with the same symbol has even length, or equivalently, each symbol appears once at an even index and once at an odd index. This parity condition was first proved necessary by Nagy. The following simpler combinatorial proof is due to Rademacher and Toeplitz.

Lemma: *Every pair of generic closed curves that intersect only transversely intersect at an even number of points.*

Proof: Let α and β be a transverse pair of generic closed curves, directed arbitrarily. Imagine a point moving around α. Each time this point crosses β, the winding number of β around that point changes by 1, and therefore changes from even to odd or vice versa. The moving point starts and ends in the same face of β, so its winding number change parity an even number of times.

Now let X be a string of length 2n, in which each of the n unique symbols appears twice.

Lemma: *If X is the Gauss code of a planar curve, then every substring of X that starts and ends with the same symbol has even length.*

Proof: Let γ be a planar closed curve. Smoothing γ at any vertex produces two subcurves α and β. Up to a cyclic shift (reflecting a change of basepoint), the Gauss code for γ can be written as $axay$, where a is the label of the vertex, substring x encodes the crossings along α, and string y encodes the crossings along β. Each self-intersection point of α is encoded in x twice, and the other symbols of x encode the intersections between α and β We conclude that x has even length, which completes the proof.

We can test this parity condition in $O(n^2)$ time by brute force, but in fact, there is a simple linear-time algorithm. Given the Gauss code X, we can define a directed graph $G(X)$, which I'll call the *Nagy graph* of X, as follows:

- The vertices of $G(X)$ correspond to the n distinct symbols in X.
- The edges of $G(x)$ correspond to (cyclic) substrings of X with length 2, alternately directed forward and backward. That is, the Nagy graph contains a forward edge $x_i \to x_{i+1}$ for every even index i and a backward edge and $x_{i+1} \to x_i$ for every odd index i. For example, the Nagy graph of the string `abcdefgchaigdjkhbifejk` contains the following edges:

$$a{\to}b{\leftarrow}c{\to}d{\leftarrow}e{\to}f{\leftarrow}g{\to}c{\leftarrow}h{\to}a{\leftarrow}i{\to}g{\leftarrow}d{\to}j{\leftarrow}k{\to}h{\leftarrow}b{\to}i{\leftarrow}f{\to}e{\leftarrow}j{\to}k{\leftarrow}a$$

Let me emphasize (despite the figure below) that the Nagy graph of a string is an *abstract graph*, which may or may not be planar.

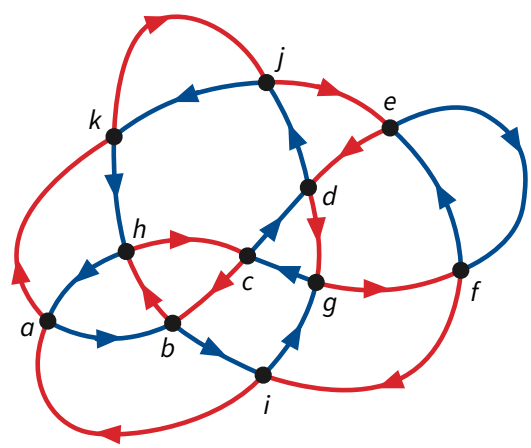

Figure 5: The Nagy graph of Gauss code `abcdefgchaigdjkhbifejk`; compare with Figure 6.

Now imagine a point moving around the Nagy graph $G(X)$, alternately traversing edges forward and backward in the order they appear in X. Whenever the point passes through a vertex of $G(X)$, it either traverses two inward edges $u{\to}v{\leftarrow}w$ (one forward and one backward) or two outward edges $u{\leftarrow}v{\to}w$ (one backward and one forward). The parity condition implies that if we leave any vertex v along a forward edge $v{\to}w$, we will next enter v along a backward edge $x{\leftarrow}v$ and v vice versa. In fact, these two conditions are equivalent.

Lemma: *A Gauss code X satisfies the parity condition if and only if every vertex of its Nagy graph $G(X)$ has in-degree 2 and out-degree 2.*

If a Gauss code does *not* satisfy the parity condition, then its Nagy graph will contain at least one vertex with in-degree 0 and out-degree 4, and an equal number of vertices with in-degree 4 and out-degree 0.

We can easily construct the Nagy graph and check the degree of each vertex in $O(n)$ time, where X is the length of the given Gauss code. (There are simpler algorithms to check the parity condition in $O(n)$ time, but we'll need the Nagy graph later, so we might as well build it now.)

Gauss also observed that the sequences abcadcedbe and abcabdecde satisfy his parity condition but cannot be realized by planar curves, so the parity condition is not sufficient. Tait later gave a third example abcadebdec.

7.4 Dehn's non-crossing condition

About 100 years after Gauss, Dehn [1] described *das Gaussische Problem der Trakte* and proposed an algorithm to solve it. Dehn observed that smoothing every vertex of a curve *to keep the curve connected* results in a simple closed curve that *touches* itself at every vertex. The same closed curve γ can have several different connected smoothings.

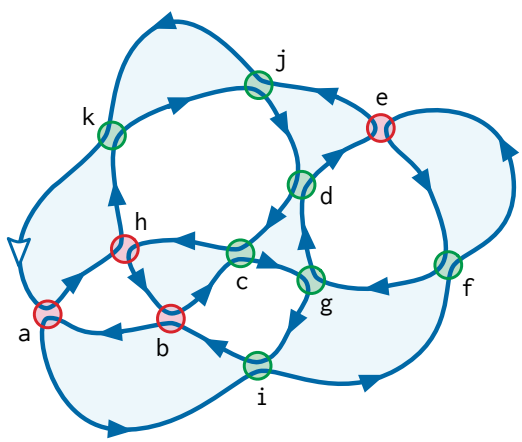

Figure 6: A smoothed curve with Dehn code ahkjdchbcgibaifefgdejk

Lemma: *Every connected smoothing of a planar curve γ is an Euler tour of the Nagy graph $G(X(\gamma))$, and vice versa.*

Proof: Consider two consecutive edges $u \to v \to w$ of γ (*not* edges of $G(X(\gamma))$). Imagine smoothing the vertices of γ one at a time. Each smoothing reverses one of the subcurves from the smoothed vertex to itself. The edges uv and vw are reversed by all these same smoothing *except* at their common vertex x. Thus, exactly one of the two subpaths $u \to v$ or $v \to w$ is revised in the final Dehn smoothing. It follows that every Dehn smoothing of γ is an Euler tour of $G(X(\gamma))$.

On the other hand, the edges incident to any vertex of $G(X(\gamma))$ alternative in, out, in, out in cyclic order. Thus, every Euler tour of $G(X(\gamma))$ touches itself at every vertex, but never crosses itself. It follows that every Euler tour of $G(X(\gamma))$ is a connected smoothing of γ.

Lemma: *A Gauss code X is realized by a planar curve if and only if its Nagy graph $G(X)$ has a planar embedding in which at least one (and therefore every) Euler tour is weakly simple.*

Proof: An Euler tour of a planar graph can cross itself only at vertices. If X is the Gauss code of a planar curve γ, the edges incident to any vertex of $G(X)$, embedded on top of γ in the obvious way, alternate in-out-in-out in cyclic order, which makes a crossing at that vertex impossible.

On the other hand, suppose $G(X)$ has a planar embedding with a weakly simple Euler tour. Then the edges incident to each vertex in that embedding must alternate in-out-in-out in cyclic order. The original Gauss code X defines an *undirected* Euler tour U of $G(X)$, which traverses edges of $G(X)$ alternately forward and backward. U crosses itself at every vertex of $G(X)$. It follows immediately that U is a closed curve with Gauss code X.

7.5 Tree-onion figures

Dehn described a symbolic algorithm to test his non-crossing condition in terms of the sequence of *self-touching* points in order along the smoothed curve $\tilde{\gamma}$. Just like Gauss codes, this sequence contains exactly two occurrences of every symbol. To distinguish this sequence from the Gauss code of a curve, I'll refer to this new string as a *Dehn code*. We can similarly define the *Dehn diagram* of $\tilde{\gamma}$ as a cycle of $2n$ vertices, corresponding to the labels in the Dehn code, plus chords connecting identical labels.[3]

Dehn observed that the Dehn diagram of any connected smoothing $\tilde{\gamma}$ of any planar curve γ is a *planar* graph; that is, we can embed some of the chords of the diagram inside the circle and the rest outside the circle, so that no pair of chords intersects. Specifically, if we perturb $\tilde{\gamma}$ slightly into a simple curve, the neighborhood of each vertex either has connected intersection with the interior of $\tilde{\gamma}$ or connected intersection with the exterior of $\tilde{\gamma}$. These vertices correspond to inner and outer chords, respectively.

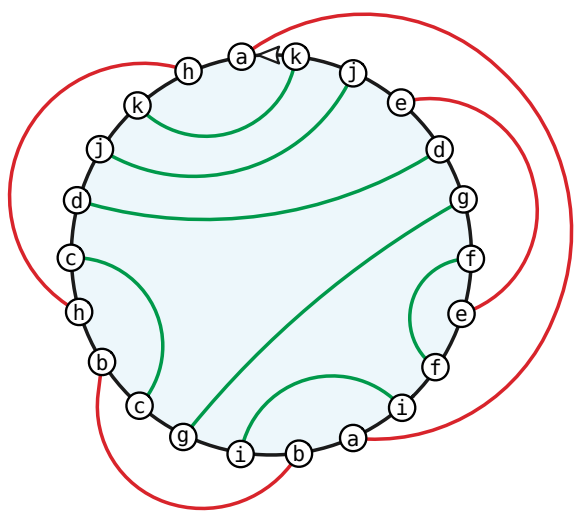

Figure 7: A planar Dehn diagram for the Dehn code `ahkjdchbcgibaifefgdejk`; compare with the previous figure!

Dehn playfully referred to these planar diagrams as "Baum-Zwiebel Figuren" ["tree-onion diagrams"] and their corresponding Gauss codes as "Baum-Zwiebel Reihen" ["tree-onion strings"]. Tree onions, also known as walking onions or Egyptian onions, are onion cultivars that grow clusters of small bulbs at the top of the stem, where other *Allium* species have flowers. The

[3]The terms "Dehn code" and "Dehn diagram" are nonstandard.

chords of a tree-onion figure (loosely) resemble clusters of onion layers. Coincidentally(?), the dual graph of the inner chords (or the outer chords) of any tree-onion figure is a tree.[4]

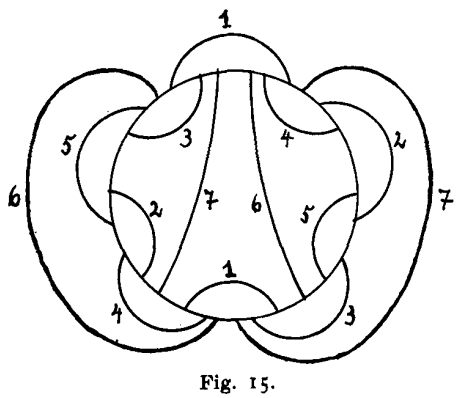

Figure 8: A tree-onion figure [Dehn 1936]

Figure 9: Tree onion bulblets. [Kurt Stüber 2004, CC BY-SA 3.0, via Wikimedia Commons]

(Petersen [9] used similar diagrams in 1891 to study abstract regular graphs. Consider a connected 4-regular graph G with n vertices. Petersen defined a "stretched graph" by representing any Euler tour of G as a cycle of length $2n$, with additional edges connecting the two occurrence of each vertex of G. If we alternately color the edges this cycle red and blue, every vertex of G is incident to two edges of each color. Thus, every 4-regular graph can be decomposed into two 2-factors. Applying Petersen's construction to any curve, as an Euler tour of its image graph, recovers the forward and backward cycles in the curve's Nagy graph.)

7.6 Bipartite interlacement

Read and Rosenstiehl [6] observed that Dehn's planarity condition can verified efficiently by constructing yet another graph, called the *interlacement graph* of the Dehn code. The interlacement graph has n vertices, one for each symbol, and an edge between any two symbols x and y whose

[4]Tree-onion figures are also closely related to tree-cotree decompositions of planar maps. Specifically, there is a bijection between tree-cotree decompositions of a planar map Σ and tree-onion figures of non-crossing Euler tours of the medial map $\Sigma^\times$. (Don't worry; those words will make sense soon.) So it's really tempting to refer to the partition of inner and outer chords in a tree-onion figure as a *coonion-onion decomposition*.

Figure 10: Egyptian tree onion. From *Child's Rare Flowers, Vegetables & Fruits* (1894)

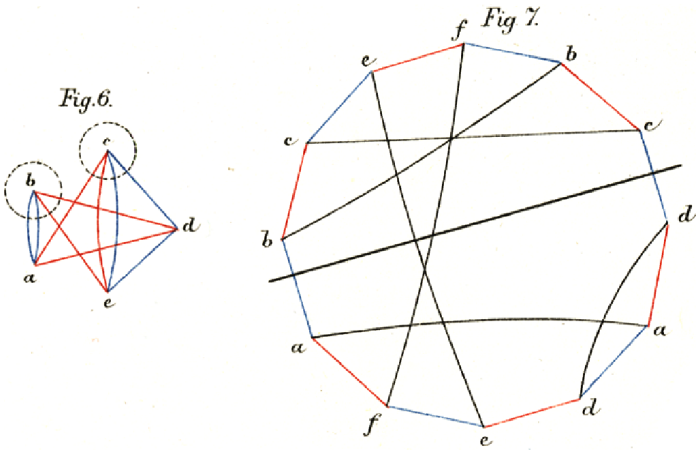

Figure 11: Petersen's diagram of an Euler tour (Petersen 1891)

appearances are interlaced $x \ldots y \ldots x \ldots y$ in the Dehn code. Partitioning the chords of a Dehn diagram into pairwise disjoint inner and outer chords is equivalent to partitioning the vertices of the interlacement graph into two independent sets. In other words, a Dehn diagram is planar if and only if its interlacement graph is bipartite.

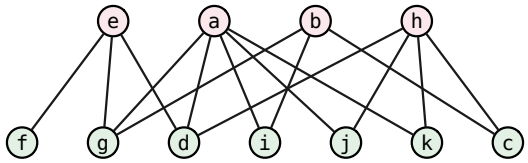

Figure 12: The bipartite interlacement graph for the Dehn code `ahkjdchbcgibaifefgdejk`

7.7 Recrossing

To complete his algorithm, Dehn observed that we can transform the Dehn diagram of any Euler tour of $G(X)$ into a 4-regular graph by replacing each chord with a pair of crossing chords with a crossing, as shown below. Assuming the interlacement graph of the Dehn code is bipartite, the corresponding Dehn diagram is planar, so this recrossing process yields a single closed curve consistent with our original Gauss code X.

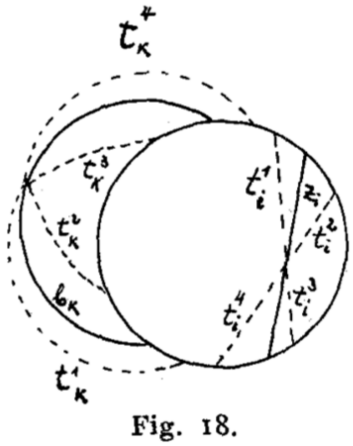

Figure 13: Building a closed curve from a tree-onion-diagram (Dehn 1936)

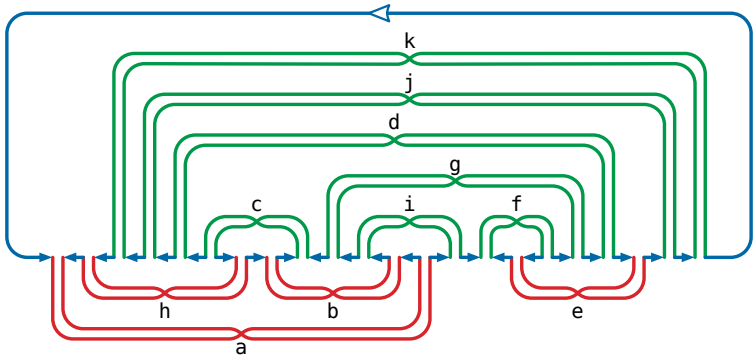

Figure 14: A planar curve consistent with the original Gauss code `abcdefgchaigdjkhbifejk`

Putting all the pieces together, we conclude:

Theorem: *A Gauss code X can be realized by a planar closed curve if and only if at least one (and therefore every) Euler tour of $G(X)$ has a bipartite interlacement graph.*

7.8 Algorithm summary

Theorem: *Given a Gauss code X of length $2n$, we can either construct a planar curve consistent with X or correctly report that no such curve exists, in $O(n^2)$ time.*

Proof: The algorithm proceeds as follows:

1. Construct the Nagy graph $G(X)$ of X. This can be done in $O(n)$ time by brute force. If any vertex of $G(X)$ has no outgoing edges, halt and report that X is not realizable. Otherwise, $G(X)$ is Eulerian, and thus X satisfies Gauss's parity condition.
2. Compute any Euler tour of $G(X)$. This can be done in $O(n)$ time using the well-known algorithm of Hierholzer (not Euler!).
3. Construct the Dehn code $\tilde{X}$ of this Euler tour. This can be done in $O(n)$ time by brute force.
4. Construct the interlacement graph $I(\tilde{X})$ of $\tilde{X}$. This can be done in $O(n^2)$ time by brute force.
5. Verify that the interlacement graph is bipartite. The interlacement graph has n vertices and at most $O(n^2)$ edges, so we can verify bipartiteness in $O(n^2)$ time uysing whatever-first search. If the interlacement graph is not bipartite, halt and report that X is not realizable.
6. Build a tree-onion figure for $\tilde{X}$ from any partition of nodes in the interlacement graph. This can be done in $O(n)$ time by brute force.
7. Transform the plane Dehn diagram into a 4-regular plane graph by replacing each chord with a pair of crossing chords, as shown in Figures 13 and 14. This can be done in $O(n)$ time by brute force; the result is a closed curve consistent with our original Gauss code X.

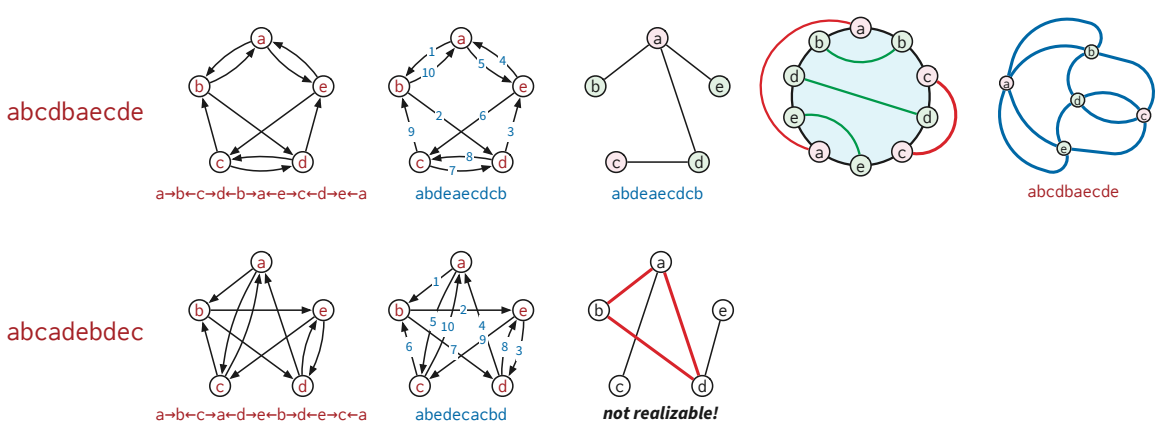

Figure 15: Two examples of Dehn's Gauss-code algorithm in action

7.9 Faster! Faster!

There are several linear-time algorithms to test the interlacement condition *without* explicitly constructing the interlacement graph, but we're out of time.

7.10 ...and the Aptly Named Yadda Yadda

- Tait-Dowker-Thistlethwaite codes
- Pile of twin stacks algorithm
- Left-right graph planarity test

7.11 References

1. Max Dehn. Über kombinatorishe Topologie. *Acta Math.* 67:123–168, 1936.
2. Clifford H. Dowker and Morwen B. Thistlethwaite. Classification of knot projections. *Topology Appl.* 16(1):19–31, 1983.
3. Hubert de Fraysseix and Patrice Ossona de Mendez. A short proof of a Gauss problem. *Proc. 5th Int. Symp. Graph Drawing*, 230–235, 1997. Lecture Notes Comput. Sci. 1353, Springer.
4. Carl Friedrich Gauß. Nachlass. I. Zur Geometria situs. *Werke*, vol. 8, 271–281, 1900. Teubner. Originally written between 1823 and 1840.
5. Carl Hierholzer. Über die Möglichkeit, einen Linienzug Ohne Wiederholung und ohne Unterbrech nung zu umfahren. *Math. Ann.* 6:30–32, 1873.
6. Louis H. Kauffman. Gauss codes, quantum groups and ribbon Hopf algebras. *Rev. Math, Phys.* 5(4):735–773, 1993.
7. Louis H. Kauffman. Virtual knot theory. *Europ. J. Combin.* 20(7):663–691, 1999. arXiv:math/9811028.
8. Julius v. Sz. Nagy. Über ein topologisches Problem von Gauß. *Math. Z.* 26(1):579–592, 1927.
9. Julius Petersen. Die Theorie der regulären graphs. *Acta Math.* 15:193–220, 1891. Yes, really, "graphs" not "Graphen".
10. Hans Rademacher and Otto Toeplitz. On closed self-intersecting curves. *The Enjoyment of Mathematics: Selections from Mathematics for the Amateur*, chapter 10, 61–66, 1990. Dover Publ. Originally published by Princeton Univ. Press, 1957.
11. Ronald C. Read and Pierre Rosenstiehl. On the Gauss crossing problem. *Combinatorics*, 843–876, 1976. Colloq. Math. Soc. János Bolyai 18, North-Holland. Modern description of Dehn's solution to the Gauss code problem.
12. Pierre Rosenstiehl and Robert E. Tarjan. Gauss codes, planar Hamiltonian graphs, and stack-sortable permutations. *J. Algorithms* 5(3):375–390, 1984. Linear-time implementation of Dehn's solution to the Gauss code problem.
13. Peter Guthrie Tait. On knots I. *Trans. Royal Soc. Edinburgh* 28(1):145–190, 1876–7.