COMBINATORICS AND TOPOLOGY OF CURVES AND KNOTS

by

Bailey Ann Ross

A thesis
submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Mathematics
Boise State University

May 2010

BOISE STATE UNIVERSITY GRADUATE COLLEGE

DEFENSE COMMITTEE AND FINAL READING APPROVALS

of the thesis submitted by

Bailey Ann Ross

Thesis Title:

Combinatorics and Topology
of Curves and Knots

Date of Final Oral Examination: 8 April 2010

The following individuals read and discussed the thesis submitted by student Bailey Ann Ross, and they evaluated her presentation and response to questions during the final oral examination. They found that the student passed the final oral examination.

Jens Harlander, Ph.D.	Chair, Supervisory Committee
Andrés Caicedo, Ph.D.	Member, Supervisory Committee
Uwe Kaiser, Ph.D.	Member, Supervisory Committee

The final reading approval of the thesis was granted by Jens Harlander, Ph.D., Chair, Supervisory Committee. The thesis was approved for the Graduate College by John R. Pelton, Ph.D., Dean of the Graduate College.

ABSTRACT

The genus of a graph is the minimal genus of a surface into which the graph can be embedded. Four regular graphs play an important role in low dimensional topology since they arise from curves and virtual knot diagrams. Curves and virtual knots can be encoded combinatorially by certain signed words, called Gauss codes and Gauss paragraphs. The purpose of this thesis is to investigate the genus problem for these combinatorial objects: Given a Gauss word or Gauss paragraph, what is the genus of the curve or virtual knot it represents?

TABLE OF CONTENTS

ABSTRACT iv

LIST OF TABLES vii

LIST OF FIGURES viii

1 Background and History 1

2 Graphs and Surfaces 4

2.1 Definition of Graph, Diagram, and Digraph 4

2.2 Definition of Walk, Simple Walk, and Euler Walk 4

2.3 Definition of Graph Embedding 6

3 Combinatorial Descriptions of Virtual Knots 7

3.1 Gauss Codes 7

3.2 Gauss Paragraphs 9

3.2.1 Converting a Gauss Paragraph into a Gauss Code 12

3.3 Definition of the Carter Surface 12

3.3.1 The Carter Surface Is a Manifold 14

3.3.2 Carter Surfaces Associated with Gauss Codes and Gauss Paragraphs . 16

3.4 Labeled Oriented Circles 17

4 Applying the Carter Surface to the Genus Problem 21

4.1 Definition of the Genus of a Graph 21

4.2 Current Status on Genus Problem 22

4.3 The Genus of a Carter Surface 25

4.3.1 The Carter Surface Generalized to Any Graph Diagram 27

4.4 Genus of Carter Surface in Linear Time 28

4.5 How Good Is the Carter Surface as an Upper Bound? 28

5 Altering a Given Diagram in a Neighborhood 31

6 A Planarity Test for Gauss Codes 34

6.1 Computations from the Gauss Code $\alpha_i(w)$ and $\beta_{ij}(w)$ 34

6.2 Definition Alexander Numbering 34

6.3 Theorem 1 from Grant Cairns and Daniel M. Elton's Paper 35

6.4 Examples of α_i and β_{ij} 40

REFERENCES 55

LIST OF TABLES

6.1 Example of Computing S_i, $\overline{S}_i$, S_i^{-1}, and α_i. 41

6.2 Example of Computing $\overline{S}_i \cap S_1^{-1}$, $\overline{S}_i \cap S_2^{-1}$, and $\overline{S}_i \cap S_3^{-1}$. 41

6.3 Computing α_i and β_{ij} for Gauss code: $\{1^+2^+3^+1^-2^-3^-\}$, Genus = 1 42

6.4 Computations for Gauss code: $\{1^+2^-3^+4^+2^+1^-4^-3^-\}$, Genus = 1 42

6.5 Computations for Gauss code: $\{1^+2^-3^-4^+1^-4^-2^+3^+\}$, Genus = 2 42

6.6 Computations for Gauss code: $\{1^+2^+1^-2^-3^+4^+3^-4^-\}$, Genus = 2 43

6.7 Computations for Gauss code: $\{1^+3^-4^+1^-2^+4^-5^+2^-3^+5^-\}$, Genus = 2 43

6.8 Computations for Gauss code: $\{1^+2^-3^-4^-5^+6^+1^-5^-2^+3^+4^+6^-\}$, Genus = 2 . . 44

6.9 Computations for Gauss code: $\{1^+2^-3^-4^-5^+6^+1^-5^-2^+3^+4^+6^-\}$, Genus = 2 . . 44

LIST OF FIGURES

2.1 a) The diagram associated with a multigraph G defined by $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{e_1 = \{v_1, v_4\}, e_2 = \{v_2, v_4\}, e_3 = \{v_3, v_4\}_1, e_4 = \{v_3, v_4\}_2\}$. b) The digraph associated with graph G defined by $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{e_1 = (v_1, v_4), e_2 = (v_4, v_2), e_3 = (v_4, v_3), e_4 = (v_3, v_4)\}$. In this case, coordinate order matters so we use ordered pairs instead of set notation. The edge is oriented starting at the first coordinate and ending at the second coordinate. 5

2.2 A graph embedded in a surface of genus one. The black represents a sphere; the red represents the handle attached to the sphere; and, the blue represents the embedded graph. Later we will see that this is the least number of handles needed to embed this particular graph. 6

3.1 a) The vertex v_i with adjacent "in" edges and adjacent "out" edges. b) The vertex v_i with edges oriented with two "in" edges opposite each other and two "out" edges opposite each other. It is easy to see that this orientation does not happen when we always walk straight. 8

3.2 The right-hand rule: The W represents the edge we are beginning to walk starting at the ith vertex. If we begin walking an edge at i, and there is another edge beginning to our right, then we replace i with i^- in our walk. If we begin walking an edge, and there is another edge beginning to our left, then we replace i with i^+ in our walk. 9

3.3 The Gauss code written from the planar diagram used in Section 3.3.1. 9

3.4 a) A virtual knot labeled with its Gauss paragraph. b) The right-hand rule, which assigns a positive or negative sign to a real crossing. The correlation is that a crossing is positive if you walk on the over-crossing edge leaving vertex i and there is an under-crossing edge to your left that is also leaving vertex i. A crossing is negative if you walk on the over-crossing edge leaving i and there is an under-crossing edge to your right that is also leaving i. 11

3.5 There is always an edge to the left of a designated edge at every vertex of order 4. This diagram shows that there are four possible left turns and if all have been used, then each edge is used twice, once in the positive direction and once in the negative direction. 13

3.6 The dotted lines represent the edges we choose to glue in a face of Carter surface M using the "turn left" rule. In this case, there are three cycles and so we will glue in three faces. The gray arrows show which direction to follow in order to always be turning left. Each edge gets used once in the positive direction and once in the negative direction. 14

3.7 The vertices are embedding into disks, which are represented by blue circles. . 15

3.8 The lighter blue strip represent bands glued to disks. When the bands are projected into the plane, we can see that there is overlap between some of them. We should think of these bands as one going over or under the other in three space. 15

3.9 The green represents the cycle if one were to begin to the left of a band and continue walking the band on the left. This cycle is the boundary to which one disk is glued. We could continue this process for the entire diagram. 16

3.10 An example of an LOC with six vertices and six edges. 18

3.11 The classical crossings associated with edges from Figure 3.10. 19

3.12 The final virtual knot diagram drawn from Figure 3.10. The light blue circles note where two edges overlap in a virtual crossing. 20

4.1 A planar diagram of a graph drawn so that edges cross over each other on the left. When we try to draw the graph so that no edges cross over (if this is possible, then we have a graph that can be embedded in a sphere), we may draw as far as the middle diagram and not be able to place the remaining edges of the graph. The dotted lines in the right diagram represent the edges that must be placed so that they cross over other edges. The last diagram shows that this particular graph diagram is not drawn in a planar way, since there are embedded edges that are not disjoint. This result does not imply that a planar diagram for the graph does not exist. 22

4.2 a) All pieces are homeomorphic to disks. The dotted line represents where the diagram was removed from S. b) Some pieces are not homeomorphic to disks, which indicates the embedding is not cellular. 23

4.3 Beginning with a diagram, writing down the vertices and edges of the graph, and then drawing a new diagram for the graph shows that we can have two different surface embeddings related to the same graph. We will always indicate, usually by drawing circles, where interior points of edges intersect, which means in the diagram these points are not vertices. When the graph diagram is drawn on a surface with a high enough genus, these interior point intersections vanish. For a nonplanar graph, it is not possible to draw a planar diagram of the graph without these crossings. 25

4.4 If we are given a diagram, we write the vertices and edges of the given graph diagram, and we draw a new diagram of the graph, then we can see that the Carter surface is not of the same genus as the genus of the graph. The diagram on the left has a Carter surface with genus one and the diagram on the right has a Carter surface of genus zero. 26

4.5 a) When a vertex has valency of four, we described the faces glued in by choosing edges in order by turning left. b) When a vertex has an even valency of six or more, we still have a well-defined turn left. c) When a vertex has an odd number of edges, there is still a well-defined turn left. 27

4.6 The given diagram has a Carter surface of genus one. 29

4.7 Place the nonplanar edges and then orient them so that the walk is always straight through at every vertex. 30

4.8 The different colors show that by using the turn left rule there will be eight different faces glued to produce the Carter surface. 30

5.1 The "R" represents a strand we color red to distinguish one circuit from the other. How to alter the given diagram: a) Inside the neighborhood (gray), the surface is the same as the surface of a sphere; outside the neighborhood, the Carter surface has any finite number of handles. b) Cut the two edges so that there are four "open-ended" edges. c) Insert a new vertex and connect it to the four "open-ended" edges. d) The "R" labels are forced to the second circuit, so now we have one single circuit. 32

5.2 a) The dotted lines represent the way in which faces are glued. Inside the neighborhood, the dotted lines represent the exact way in which faces will be glued; however, outside the neighborhood, the dotted lines represent general patterns that will be followed when gluing in edges. b) Inside the neighborhood, the dotted lines represent the exact way in which faces will be glued according to the placement of the new vertex and edges. Outside the neighborhood, the dotted lines represent how the new diagram has not changed from the old diagram, so the general pattern that will be followed when gluing in edges remains the same. It is clear in the new diagram that we have an additional vertex, two additional edges, and one additional face. . . 33

6.1 An example of an Alexander numbering for the classical trefoil knot. 35

6.2 The curve turns to the left or the right at a vertex. 36

6.3 The given Alexander numbering provides the $a_1, ..., a_m$ for $\partial(a_1 f_1 + ... + a_m f_m)$ and so γ is null-homologous. Specifically, every e_i is found exactly once after calculating $\partial(a_1 f_1 + ... + a_m f_m)$. 37

CHAPTER 1

BACKGROUND AND HISTORY

The use of topological ideas to explore various aspects of graph theory is a fruitful area of research. The origins of topological graph theory lie in the 19th century with such famous questions as the four color problem. In the 1930s, Kuratowski characterized the planarity of graphs by two obstructions, and generalized his results to higher order surfaces, which became a well-known open problem (solved by Robertson and Seymour in the 1990s) [1]. Typically, problems in topological graph theory center around the question of which graphs can be embedded in which surfaces.

The solution to many classical problems in topological graph theory is algorithmic in nature and it is natural to ask how fast (or good) these algorithms are. One example is the genus problem for graphs, which is the problem of finding the minimal genus surface into which a given graph embeds is NP-hard [1]. Questions of computational complexity have renewed interest in many problems in topological graph theory.

Graphs arise in a variety of mathematical subjects. In this thesis, we primarily focus on 4-regular graphs, that is graphs in which each vertex is contained in exactly four edges. These graphs naturally arise in low-dimensional topology. A 4-regular planar graph, where over- and under-crossing information is provided at the vertices, is a classical link diagram (knot diagram in the case of a single component), discussed in Chapter 3. Sophisticated link and 3-manifold invariants can be read from this diagram, such as presentations for the fundamental group of the link complement or various link and knot polynomials. If one provides over- and under- crossing information for some vertices but not for all, one obtains a virtual link diagram [4]. Thus, a virtual link diagram is a projection of a 4-regular graph into the plane and it is important to note that there are many different such projections for the same graph. It can be shown that virtual link diagrams define ribbon 2-sphere complements in the 4-sphere. The study of 4-regular graphs is also the study of immersed curves and goes all the way back to C. F. Gauss [8]. He noticed that curves on surfaces can be encoded by strings of signed letters, now called Gauss codes (see Chapter 3 for more details). More precisely, Gauss noticed that any string obtained from permuting the letters in the string $a_1^+a_1^-...a_n^+a_n^-$ encodes a curve on a surface, and he wondered how one can tell if it encodes a planar curve.

The 4-regular graphs can be encoded combinatorially in a variety of ways. We already mentioned Gauss codes. Labeled oriented circles, LOCs for short, provide a method for encoding 4-regular graphs with over- and under-crossing information. Typically, one asks what combinatorial properties of the LOC imply planarity. A Gauss code or LOC always allows for the construction of a planar projection of the 4-regular graph defined, and hence of a surface into which the graph embeds, but it might not be the one of minimal genus. In [3], Harlander and Rosebrock give conditions for the LOC that imply non-planarity of the graph; and in [5], Cairns and Elton characterize the Gauss codes that define curves that are planar. The original motivation for this thesis was to find conditions on an LOC that characterize planarity.

The main point of this thesis is to study the genus problem for Gauss codes, Gauss paragraphs, and LOCs. Our work is structured as follows. In Chapter 2, we provide basic definitions about graphs and graph embeddings into surfaces. In Chapter 3, we discuss simple closed curves, links, and virtual links, and various combinatorial objects associated with them. These are Gauss codes, Gauss paragraphs, and LOCs. Curves, links, and virtual links all come with surfaces and so do the combinatorial objects. These surfaces, called Carter surfaces (named after Scott Carter [8]), are also described in Section 3. The genus problem in this context can be formulated in the following way: Given a Gauss code, Gauss paragraph, or LOC, what is the genus of the associated Carter surface?

In Chapter 4, we give a brief discussion of the classical genus problem for graphs. We can apply the Carter-surface construction to a planar projection of any graph and thus obtain an upper bound on the genus of the graph. However, the surface constructed depends on the chosen projection and not just on the graph.

Chapter 5 is technical in nature. We show that a planar projection of a 4- regular graph can be locally altered and changed to a projection with a single component without changing the genus of the Carter surface. So, from a genus point of view, one can reduce the study of virtual links to the study of virtual knots, or put differently, reduce the study of Gauss paragraphs to the study of Gauss codes. In Chapter 6, we review the beautiful work of Cairns and Elton [5]. They construct invariants α_i and β_{ij} from a Gauss code and prove that the curve defined by the code is planar if and only if the invariants all vanish. In fact, they show that the Carter surface associated with the code is a sphere if and only if all the invariants vanish. The question arises whether the α_i and β_{ij} contain more information about the Carter surface. We go through the virtual knot table of Jeremy Green and Dror Bar-Natan and compute some of the invariants for virtual knots [9]. Our findings support the idea that the α_i and β_{ij} detect the genus of the Carter surface of a given Gauss code.

CHAPTER 2

GRAPHS AND SURFACES

In this chapter, we provide basic definitions concerning graphs and graph embeddings.

2.1 Definition of Graph, Diagram, and Digraph

A **graph** G or Γ is a pair of sets, a set $V(G)$ of vertices and a set $E(G)$ of edges, where $V(G)$ is nonempty and $E(G)$ is a set consisting of two element subsets of $V(G)$; see Figure 2.1a [1]. In general, there are instances when two or more edges are needed to connect two vertices. In such a situation, we call the graph a **multigraph** [2]. We can represent this information in our set $E(G)$ by using indices to indicate the different edges. A **graph diagram** is a planar representation (see Section 2.3 and Section 4.1) of sets $V(G)$ and $E(G)$, where each vertex is a point; and for all the pairs of vertices that are the endpoints of an edge, we connect the two respective points by a line. For simplicity, we call these points "vertices" and the lines we call "edges." A graph diagram is a **directed graph** or **digraph** if an edge e is given an arrow to indicate the direction the edge travels from one vertex of e to the other; see Figure 2.1b [2].

2.2 Definition of Walk, Simple Walk, and Euler Walk

A **walk** in a graph G is a finite sequence of vertices $v_0, ..., v_k$ and edges $e_1, ..., e_k$ of G: $v_0e_1v_1e_2v_2...v_{k-1}e_kv_k$, where the endpoints of e_i are v_{i-1} and v_i for each $i = 1, 2, ..., k$. A walk is called **simple** if no edges are repeated in the sequence. When a walk begins and ends with

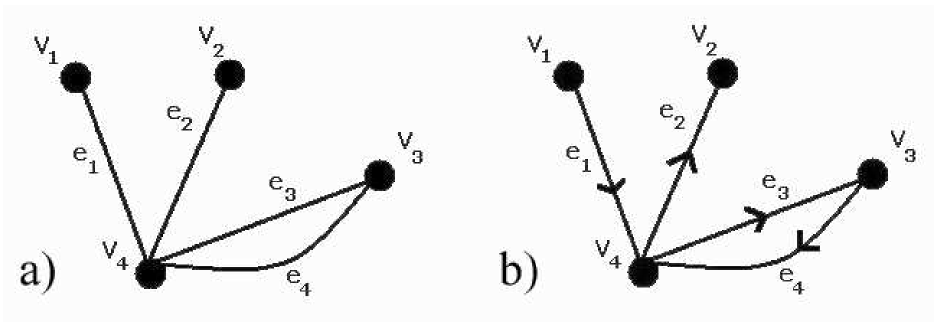


Figure 2.1: a) The diagram associated with a multigraph G defined by $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{e_1 = \{v_1, v_4\}, e_2 = \{v_2, v_4\}, e_3 = \{v_3, v_4\}_1, e_4 = \{v_3, v_4\}_2\}$. b) The digraph associated with graph G defined by $V(G) = \{v_1, v_2, v_3, v_4\}$ and $E(G) = \{e_1 = (v_1, v_4), e_2 = (v_4, v_2), e_3 = (v_4, v_3), e_4 = (v_3, v_4)\}$. In this case, coordinate order matters so we use ordered pairs instead of set notation. The edge is oriented starting at the first coordinate and ending at the second coordinate.

the same vertex, we call the walk a **circuit**. A **Euler walk** is a simple walk that contains every edge in a multigraph [2]. On a digraph, when we walk with an arrow, we will refer to this walking on an edge in a **positive direction**; and when we walk against an arrow, we will refer to this as walking on an edge in a **negative direction**. Consider the graph diagram in Figure 4.3. A possible Euler walk that is a circuit would be $v_2e_1v_1e_3v_2e_2v_1e_4v_2$.

Theorem 2.2.0.1 [2] *If G is a connected multigraph and every vertex of G has an even degree, then G has an Euler walk starting and ending at the same vertex.*

In this paper, we will be making observations about multigraphs where every vertex has an even degree. We will use the Theorem 2.2.0.1 later in the paper.

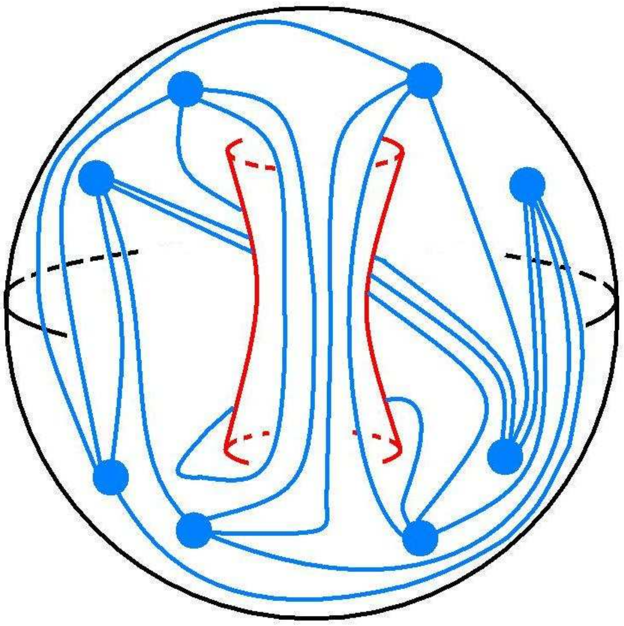


Figure 2.2: A graph embedded in a surface of genus one. The black represents a sphere; the red represents the handle attached to the sphere; and, the blue represents the embedded graph. Later we will see that this is the least number of handles needed to embed this particular graph.

2.3 Definition of Graph Embedding

A graph G is **embedded** in a topological space X if the vertices of G are distinct elements of X and every edge of G is a simple arc (in X) connecting the two elements that correspond to the two vertices that it joins in G (the edge minus the vertices of the embedded edge is disjoint from all other embedded edges and vertices). An **embedding** of G in the topological space X is a map that is an isomorphism of G with a graph G' embedded in X [1]. The planar projection of an embedded graph is a graph diagram. In this paper, we are interested in embedding G into two-dimensional surfaces; see Figure 2.3. An embedding of G in a surface is **cellular** if every face of G is homeomorphic to an open disc in $\mathbb{R}^2$ [1].

CHAPTER 3

COMBINATORIAL DESCRIPTIONS OF VIRTUAL KNOTS

In this section, we define the combinatorial objects in which we are mainly interested. These are Gauss codes, Gauss paragraphs, and labeled oriented circles (LOCs).

3.1 Gauss Codes

Let γ be a closed curve on some surface. Consider a planar projection of γ. This projection can be interpreted as a 4-regular multigraph diagram; call it G. Since G comes from a closed curve, it is easy to see that there is a single circuit that can be walked so that the edges of G are given an orientation by "walking straight" at every vertex. We can also see that this walk will be a Euler walk. A **Gauss code** or **Gauss word** is a sequence of vertices with designated superscripts of either $+$ or $-$ that encodes this specific type of Euler walk. Since G is a circuit, we may begin at any vertex and we know we will end at the same vertex. Observe how every edge is adjacent to two other edges. By 'adjacent' we mean that while walking on one edge and passing through a vertex, there is an edge immediately to the left and immediately to the right. By going straight at every vertex throughout the walk, we see that every vertex v_i is oriented with two adjacent edges "coming into v_i" and two adjacent edges "coming out of v_i"; see Figure 3.1.

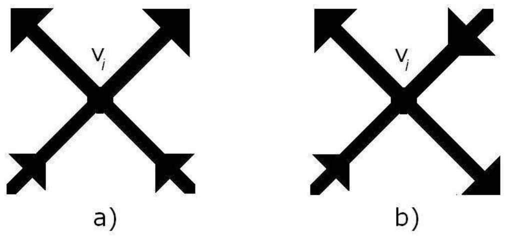


Figure 3.1: a) The vertex v_i with adjacent "in" edges and adjacent "out" edges. b) The vertex v_i with edges oriented with two "in" edges opposite each other and two "out" edges opposite each other. It is easy to see that this orientation does not happen when we always walk straight.

Enumerate the vertices $1, ..., n$. We begin encoding by writing down the Euler walk described above: $v_0e_1v_1e_2v_2...v_{k-1}e_kv_k$, where $v_j \in \{1, ..., n\}$ for $j = 1, ..., k$. Since our walk is a circuit, it does not matter at which vertex we begin and end. That means $v_0 = v_k$ in our Euler walk. We will write the Gauss code in cyclic notation; therefore, eliminate v_0 and treat the walk $e_1v_1e_2v_2...v_{k-1}e_kv_k$ as a cycle. Next, we get rid of all edges. What remains of our walk is $v_1v_2...v_{k-1}v_k$. Since we are considering a walk on a multigraph diagram, we can draw an arrow to indicate each edge's orientation. For each $i \in \{1, ..., n\}$, there will always be two edges beginning at i and two edges ending at i, so i should appear in our walk twice. When we are leaving a vertex v_j by beginning on a new edge, in our walk we write on v_j the superscript $-$ if there is another edge beginning at v_j to our right and we write on v_j the superscript $+$ if there is another edge beginning at v_i to our left. This orientation of crossings is called the right-hand rule; see Figure 3.2. Our final walk is in the form $\{a_1...a_{2n}\}$, where $a_j \in \{1^\pm, ..., n^\pm\}$ for $j = 1, ..., 2n$; see Figure 3.3 for an example. Note that the code does not depend on the planar projection chosen. In summary, a Gauss code is a word in $\{1^+, 1^-, ..., n^+, n^-\}$, obtained from permuting the letters in the word $1^+1^-...n^+n^-$. We have described how to obtain a Gauss code from a curve. We leave it to the reader to check that this process can be reversed. Given a Gauss code w, one may draw a planar projection of γ

associated with w.

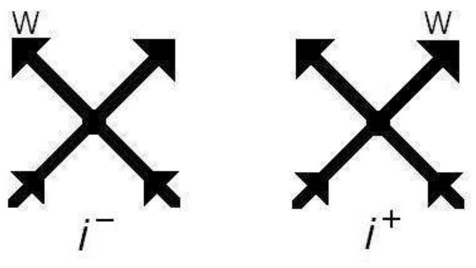


Figure 3.2: The right-hand rule: The W represents the edge we are beginning to walk starting at the ith vertex. If we begin walking an edge at i, and there is another edge beginning to our right, then we replace i with i^- in our walk. If we begin walking an edge, and there is another edge beginning to our left, then we replace i with i^+ in our walk.

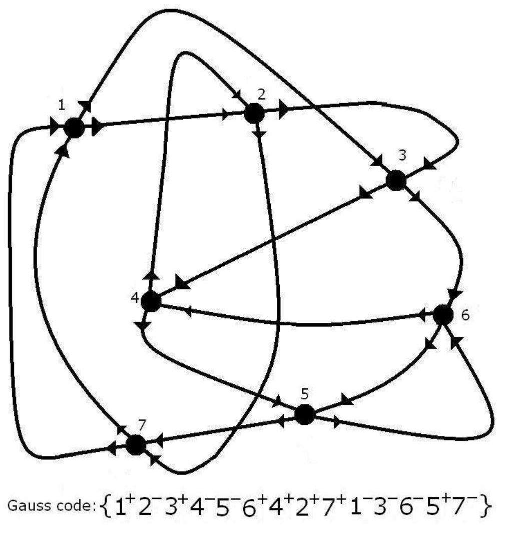

1
2
3
4
5
6
7
Gauss code: $\{1^+2^-3^+4^-5^-6^+4^+2^+7^+1^-3^-6^-5^+7^-\}$

Figure 3.3: The Gauss code written from the planar diagram used in Section 3.3.1.

3.2 Gauss Paragraphs

A **knot** is an embedding of a circle into the 3-ball. A **link** is an embedding of a collection of circles into the 3-ball. A **virtual link** is an embedding of a collection of circles into a thickened closed surface. By a 'thickened' surface we mean a space $S \times [0,1]$, where S is a surface. For technical reasons, one cones off one side, say $S \times \{0\}$, of the thickened surface. Note that the 3-ball is a thickened 2-sphere with one side coned off, so a link is also a virtual link. We are only concerned with projections of links and virtual links. Such projections are called **link** and **virtual link diagrams**. A **virtual link diagram** is a planar 4-regular graph drawn on a surface in which we have over- and under-crossing information at the vertices. Typically, one draws this diagram in the plane, which introduces **virtual crossings**, that is vertices in which we do not have over- and under-crossing information. These virtual crossings come from the handles of the surface. So, in summary, a virtual link diagram is a planar 4-regular graph, with over- and under-crossing information at some vertices (referred to as **real crossings**) but not for others (referred to as **virtual crossings**).

A **Gauss paragraph** encodes a virtual link, where each word in the Gauss paragraph is similar to a Gauss code (here we will refer to them as words). Another difference between a Gauss code and a Gauss paragraph is that the Gauss paragraph encodes over/under information while the Gauss code does not distinguish between real or virtual crossings.

Given a virtual knot projection, we begin writing the Gauss paragraph by enumerating the real crossing with $1, 2, ..., n$. We give the edges in the virtual knot projection an orientation by starting at some edge and walking straight at every vertex until we arrive back where we started. There is a choice to make when orienting the remaining circuits (components of the virtual link), and the Gauss paragraph will distinguish between the choices. Once all the edges are oriented, we can assign a $+$ or $-$ to each crossing using a variation on the right-hand rule; see Figure 3.4b. Notice that the virtual crossings are not enumerated and are not designated as $+$ or $-$. The Gauss paragraph is a set of cyclic words, each word made of letters that represent going over or under a real crossing. Since it does not matter where we begin, for convenience we begin with going over the first crossing. When we walk straight through a real crossing by going under, we write the number of the crossing only. When we walk through a crossing by going over, we write the number of the crossing with a superscript $+$ or $-$ depending on the sign of the crossing. If our walk comes to a virtual crossing, then we do not write anything and walk straight through the vertex continuing to the next crossing. We continue writing down vertices with or without superscripts until we return back to where we began right before going over the first crossing. If there is one component, then the Gauss paragraph will have one word. If there is more than one component, then we put a comma after the first cyclic word and begin a new cyclic word starting before a real crossing that has not been recorded in previous words; see Figure 3.4a for an example. **Words** of the Gauss paragraph are the cycles that represent individual components of the virtual link. **Letters** of the Gauss paragraph are the numbers that correlate to real crossings in the diagram. We leave it to the reader to check that this process can be reversed. Like the Gauss code, the Gauss paragraph is an invariant for the virtual link and does not depend on the projection.

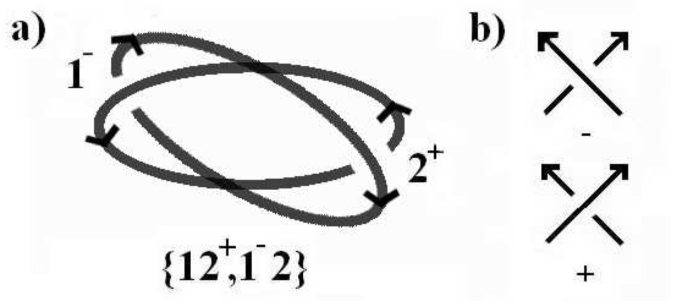


Figure 3.4: a) A virtual knot labeled with its Gauss paragraph. b) The right-hand rule, which assigns a positive or negative sign to a real crossing. The correlation is that a crossing is positive if you walk on the over-crossing edge leaving vertex i and there is an under-crossing edge to your left that is also leaving vertex i. A crossing is negative if you walk on the over-crossing edge leaving i and there is an under-crossing edge to your right that is also leaving i.

3.2.1 Converting a Gauss Paragraph into a Gauss Code

To write a Gauss code, we need a single component. There exists a construction, given below, for making a n-component virtual link v into a single component virtual knot v' by inserting $n-1$ crossings. Adding these crossings is a local move and we discuss in Chapter 5 that the Carter surfaces of v and v' will have the same genus.

If we are given a Gauss paragraph of virtual knot, then the Gauss code may be written in the following way: write the Gauss paragraph and whenever there is a letter without a superscript give this letter the superscript that is the opposite of the superscript that appears elsewhere on the same letter in the word. If we walk the virtual knot diagram associated with the Gauss paragraph, and treat all crossings as if they have no over/under information to write the Gauss code for the diagram, then we will have the same code as if we had simply filled in the Gauss paragraph with the missing signs.

If given a Gauss paragraph with n words, then we can use the following construction to write a Gauss code that associates with the Gauss paragraph: take two words that are linked by a crossing, say w_1 and w_2 with crossing a_i in common. Without loss of generality, assume w_1 has a_i and w_2 has a_i^+ or a_i^-. Rewrite w_2 cyclically so that a_i^+ or a_i^- is the last letter of w_2. We insert a crossing here by doubling the ith crossing using the following rule: a_i becomes $a_i\tilde{a}_i{}^+$ and a_i^+ becomes $a_i^+\tilde{a}_i$. If the crossing in w_2 is a_i^-, then a_i becomes $a_i\tilde{a}_i{}^-$ and a_i^- becomes $a_i^-\tilde{a}_i$. Now we insert the new w_2 that ends in $a_i^+\tilde{a}_i$ (or $a_i^-\tilde{a}_i$) directly after $a_i\tilde{a}_i{}^+$ (or $a_i\tilde{a}_i{}^-$) into w_1. We continue this process until we have a single component Gauss paragraph and then we proceed to write this paragraph as a Gauss code [7].

3.3 Definition of the Carter Surface

Closed curves, links, and virtual links all involve surfaces. In the preceding sections, we converted these topological objects into combinatorial ones. Not surprisingly, the combinatorial objects include the information to reconstruct the associated surfaces. We take the following approach: given a Gauss code, Gauss paragraph, or an LOC (to be discussed later), we first derive a diagram (i.e., a planar projection of the curve or the virtual link defined by the code), and construct a surface from the projection. This surface is independent of the particular projection chosen. (Here, by "planar projection," we mean a 4-regular graph drawn in the plane but not necessarily embedded in the plane.)

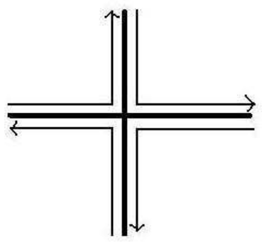


Figure 3.5: There is always an edge to the left of a designated edge at every vertex of order 4. This diagram shows that there are four possible left turns and if all have been used, then each edge is used twice, once in the positive direction and once in the negative direction.

First we will describe the original definition of a Carter surface. Given a planar projection of a connected multigraph consisting of vertices and edges where every vertex is of order four, we can create a surface by gluing in faces. Glue faces in the following way: for the first face, start at some vertex and follow an edge to another vertex, then turn left at this vertex and continue on this edge. The direction "turn left" is well defined since there is always an edge to the right, an edge to the left, and an edge directly across from the any edge connected to a vertex of order four; see Figure 3.5. Repeat this process of "turning left" at each endpoint vertex until you return back to the beginning vertex, and we call this closed path a **cycle**. If you have used each edge twice (once in the positive direction and once in the negative direction), then you are done. If you have not used each edge twice, then begin gluing in a second face by going to an edge that is unused and repeat the process of turning left. If there are no edges that are unused, then choose an edge that was only used once and begin the process, traveling in the opposite direction of that which was already used. Choosing the edges in this way gives the directions on how to glue each face, one for each cycle; see Figure 3.6. This surface is called the **Carter surface** associated with a given multigraph diagram.

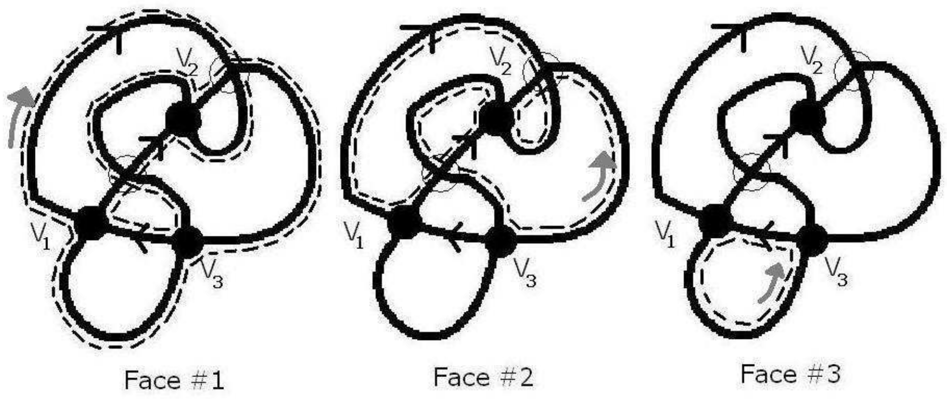


Figure 3.6: The dotted lines represent the edges we choose to glue in a face of Carter surface M using the "turn left" rule. In this case, there are three cycles and so we will glue in three faces. The gray arrows show which direction to follow in order to always be turning left. Each edge gets used once in the positive direction and once in the negative direction.

3.3.1 The Carter Surface Is a Manifold

Given a planar graph diagram we create a 2-dimensional cell complex, which we called the Carter surface. In this section, we want to make sure that this complex is indeed a surface. The following is a the construction of the Carter surface from a topological point of view.

Given a planar graph diagram, we have vertices and edges. We begin by embedding the vertices into disks. The visual image is fattening up our vertices; see Figure 3.7.

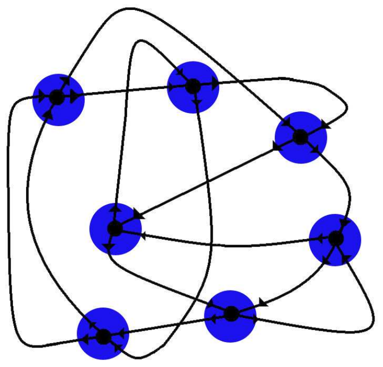


Figure 3.7: The vertices are embedding into disks, which are represented by blue circles.

Next, we embed the edges into untwisted bands (sometimes called ribbons) and glue these untwisted bands to the disks so that there is no overlap between any two bands glued to the disk; see Figure 3.8.

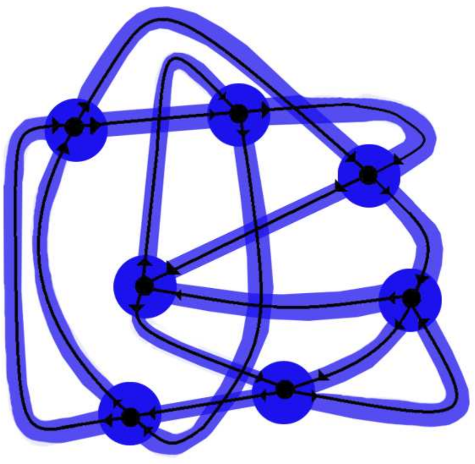


Figure 3.8: The lighter blue strip represent bands glued to disks. When the bands are projected into the plane, we can see that there is overlap between some of them. We should think of these bands as one going over or under the other in three space.

The bands glued to disks produce an orientable 2-manifold with boundary. Using the "turn left" rule previously described, we can see that each boundary is a circular component; see Figure 3.9. We glue in a disk for every boundary component to produce an orientable manifold, which is called the Carter surface. The embedding of the graph diagram is obviously cellular by construction.

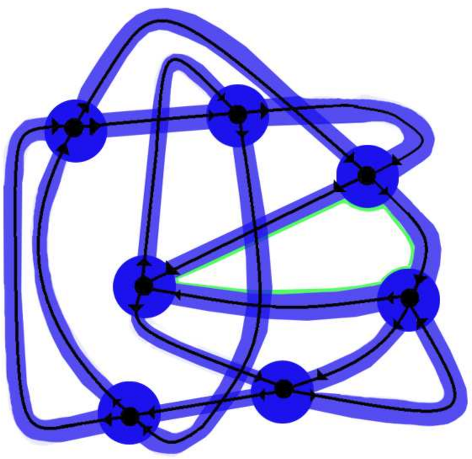


Figure 3.9: The green represents the cycle if one were to begin to the left of a band and continue walking the band on the left. This cycle is the boundary to which one disk is glued. We could continue this process for the entire diagram.

3.3.2 Carter Surfaces Associated with Gauss Codes and Gauss Paragraphs

Let w be a given Gauss code or word with $2n$ letters. We construct a surface M or $M\{w\}$ in the following way: each letter appears twice in a Gauss word, therefore there are n vertices. We have an edge connecting vertices a and b for adjacent letters ab or ba found in w, where $a \in \{i^+, i^-\}$ and $b \in \{j^+, j^-\}$, for some $i, j \in 1, ..., n$. Reminder, words are cyclic and so for the last letter of the word and the first letter of the word, there will be an edge. This construction makes $2n$ edges and the result is a planar diagram. Now we glue in faces so that each edge is used twice, once in the positive direction and once in the negative direction using the previously described "turn left" rule. The result is the Carter surface associated with the Gauss code w. The Carter surface associated with a given Gauss paragraph is constructed in a similar way, except for a difference in superscripts.

3.4 Labeled Oriented Circles

There is another way to combinatorially encode a virtual link via so called labeled oriented circles, or LOCs for short. These objects first arose in the study of higher dimensional knots, that is embeddings of the 2-sphere into the 4-sphere. This work was in part motivated by the genus question for LOCs. In [3], Harlander and Rosebrock give conditions on an LOC that imply non-planarity. For completeness, we include a discussion of LOCs in this subsection.

An **LOC** consists of vertices and edges with orientation; see Figure 3.10. Letters correlate to edges, or arcs, of the virtual knot from one under crossing to another under crossing. In other words, when walking the virtual knot on an arc labeled a, whenever we cross under another arc, we label the arc b. If we are on an arc labeled a and cross over another arc, then the label remains a. Each edge of an LOC represents a classical knot crossing. If the edge (a, b) labeled by c appears in the graph, then the arc c is passing over where arc a is changing to arc b. We place arrows on the arcs to indicate the direction we face when walking on the virtual knot. For vertices (which represent one arc going under another arc), we will always put arrows facing the same direction. The arrow on the edge of the LOC indicates which direction we are facing when we walk the over arc in relating to the under arc; see Figure 3.11. We will assume that we are working with a single LOC, where each letter appears once on an edge and once on a vertex (which is known as an injective LOC). We need this condition to have a well-defined process for drawing the virtual knot associated with the LOC.

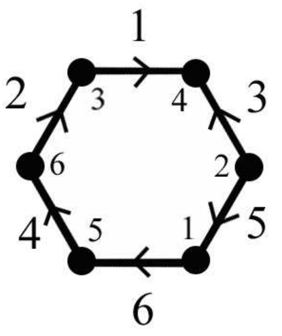


Figure 3.10: An example of an LOC with six vertices and six edges.

To draw a virtual knot diagram associated with an LOC, draw one classical crossing for each edge of the LOC. We will draw these crossing with the over arc going up and down and the under arc going left and right, making a small "t" instead of an "x." It is best to leave space between them so it is easy to connect arcs later. Each crossing correlates to a particular edge so for each crossing do the following in reference to the correlating edge of the LOC: draw two parallel arrows (marked with the same letter and pointing the same direction as the correlating edge of the LOC) on the over-crossing arc: one above and one below the crossing. Draw two parallel arrows pointing the same direction (up) perpendicular to the other two arrows already drawn: one to the left and one to the right of the crossing. Mark these arrows with letters (which correlate to the vertices of the edge in the LOC) so that the initial vertex letter is on the left and the terminating vertex letter is on the right. After we have finished this step, we see one classical crossing for every edge drawn in a square made of labeled arrows, and each letter appears on an arrow four times; see Figure 3.11.

The last step is to connect all the arcs that are labeled with arrows of the same letter and direction. We begin with an under arc that is labeled by an arrow and connect it to the over arc labeled with an arrow of the same direction and letter, then connect the remaining over arc to the remaining under arc of the same letter (these match direction automatically).

We do this for all arrows until the entire virtual knot is drawn; see Figure 3.12. Again, we leave it to the reader to check that this process can be reversed.

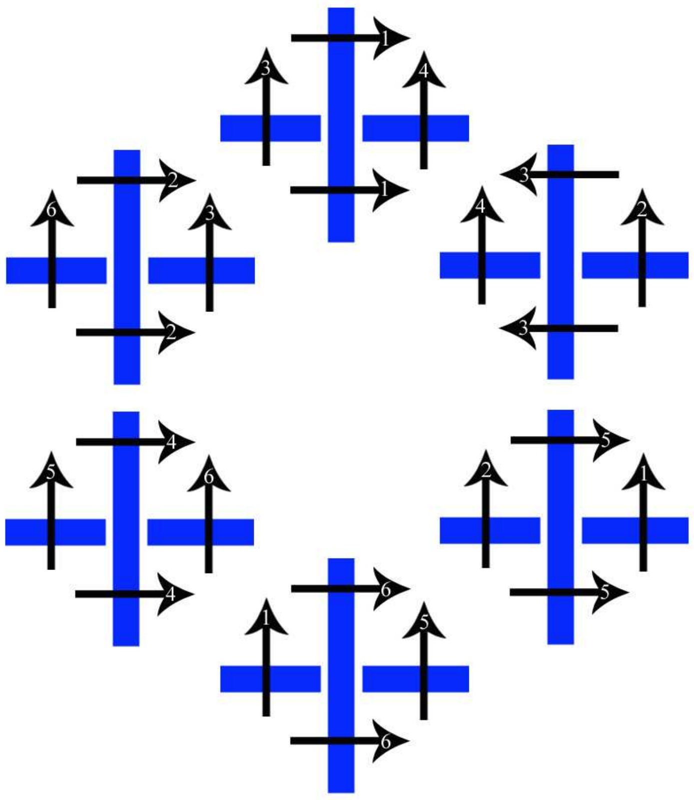


Figure 3.11: The classical crossings associated with edges from Figure 3.10.

It is easy to see that from one LOC we may draw many different virtual knot diagrams depending with which vertex we begin connecting the arcs. The only difference between the diagrams is where and how many virtual crossings appear. Since we “ignore” virtual crossings when calculating the Carter surface, we know that all virtual knot diagrams associated with the LOC have the same Carter surface. Therefore, given an LOC, we have a unique Carter surface associated with it.

There is a natural surface that arises from the virtual knot associated with an LOC. In Figure 3.12, we see that every edge in the LOC is associated with a crossing. With the labeled arrows, we see a square in which the crossing is drawn. Let every square be a face with four sides. A side of a face can be identified with another side of a face simply by following the connecting arc that passes through the side all the way to the adjacent crossing. When we do this for every side of every face, we create a surface composed of squares. This surface is dual to the Carter surface associated with the same LOC.

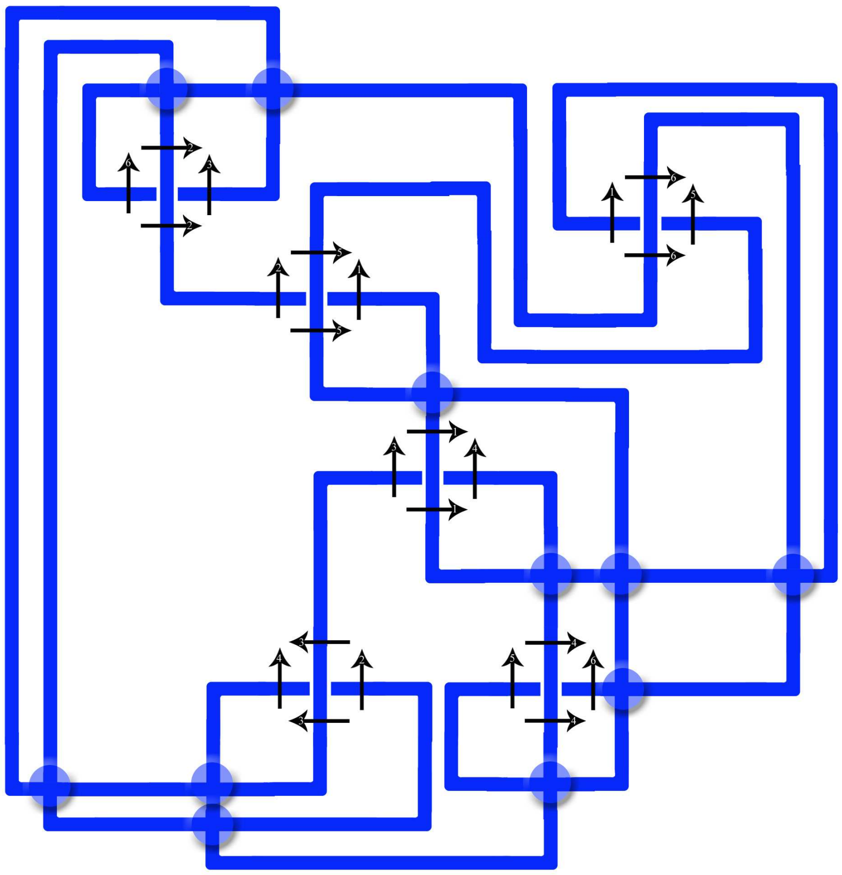


Figure 3.12: The final virtual knot diagram drawn from Figure 3.10. The light blue circles note where two edges overlap in a virtual crossing.

CHAPTER 4

APPLYING THE CARTER SURFACE TO THE GENUS PROBLEM

In this section, we discuss the genus problems for graphs: given a graph, find the lowest genus surface in which it embeds. A planar diagram of the graph allows for the construction of a Carter surface. The genus of that surface is an upper bound for the genus of the graph. Of course, this surface depends on the chosen projection, so it is not clear how good an upper bound we obtain. This makes the Carter surface a somewhat weaker tool than in the case of curves or virtual links.

4.1 Definition of the Genus of a Graph

The **genus of a graph** G, denoted $g(G)$, is the minimal integer n, such that G has an embedding into the surface of genus n. The **genus of an orientable surface** S, denoted $g(S)$, is the number of handles h added to the 2-sphere to obtain S [1]. We use g to represent genus in both cases because the meaning is obvious depending on the circumstance. We can think about the genus of a graph as the minimum number of handles that must be added to a sphere in order to embed the graph on the surface by combining the previous definitions. If $g(G) = 0$, then we call the graph **planar** and the graph can be embedded into a sphere without any intersections of interior points of edges. If $g(G) > 0$, then we say the graph is **nonplanar**. Drawing planar diagrams of graphs makes picturing the graph easier, but there are many different diagrams that are associated with the same graph; see Figure 4.3. The genus of a graph is the minimum number of handles needed and the minimum is taken over all graph diagrams. How do we draw all possible diagrams of a graph? It is clear that the more complicated the graph, the more complicated it is to calculate the genus of a graph based on diagrams. See Figure 4.1 for the process of attempting to calculate the genus of a graph. We know that the graph in Figure 4.1 is nonplanar [1].

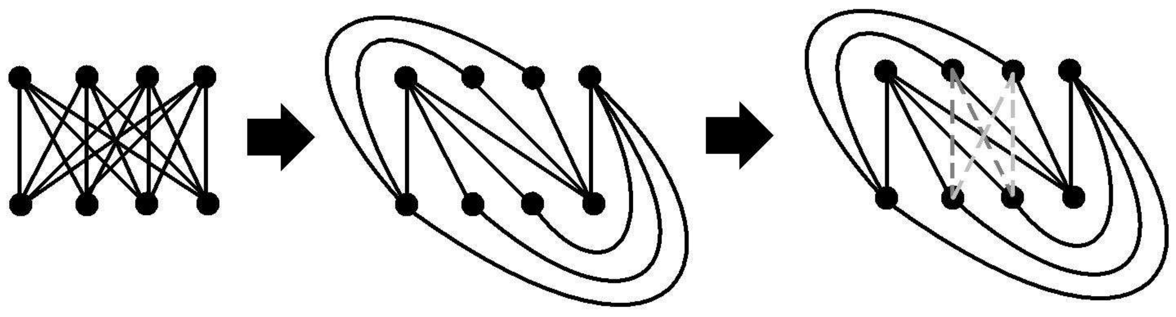


Figure 4.1: A planar diagram of a graph drawn so that edges cross over each other on the left. When we try to draw the graph so that no edges cross over (if this is possible, then we have a graph that can be embedded in a sphere), we may draw as far as the middle diagram and not be able to place the remaining edges of the graph. The dotted lines in the right diagram represent the edges that must be placed so that they cross over other edges. The last diagram shows that this particular graph diagram is not drawn in a planar way, since there are embedded edges that are not disjoint. This result does not imply that a planar diagram for the graph does not exist.

4.2 Current Status on Genus Problem

The genus problem: given a graph G and a natural number k, is $g(G) \leq k$?

The status of the general genus problem is NP-complete [1]. There has been progress on more specific aspects of the problem through multiple approaches. The most common approach is an upper bound or maximum. Obviously, once a graph is embedded in a surface, we may add as many handles as we want and still have the graph embedded in this surface. It is efficient to only consider graphs where the embedding of G is cellular, which hinders our adding superfluous handles. Let S be any surface such that there is an embedding of G in S. The removal of G from S (denoted $S - G$) cuts S into several connected pieces. The embedding of S is a **cellular embedding** of G if the connected pieces of $S - G$ are disks; see Figure 4.2a. If the pieces are disks, then there are no handles that are part of S while not containing any parts of the embedded graph; see Figure 4.2b.

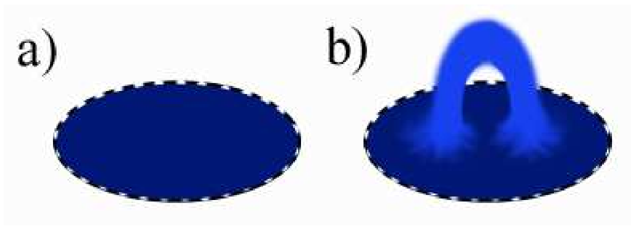


Figure 4.2: a) All pieces are homeomorphic to disks. The dotted line represents where the diagram was removed from S. b) Some pieces are not homeomorphic to disks, which indicates the embedding is not cellular.

We define an upper bound called the **maximum genus** $g_{MAX}(G)$ to be the largest integer g such that G has an cellular embedding of G in a surface of genus g [1]. It is clear the $g(G) \leq g_{MAX}(G)$ and so if we find $g_{MAX}(G)$ or an upper bound for $g_{MAX}(G)$, then we have found an upper bound for $g(G)$.

Theorem 4.2.0.1 [1] *Let G be a connected graph with v vertices and e edges. Then,*

$$g_{MAX}(G) \leq \lfloor \frac{e - v + 1}{2} \rfloor.$$

Proof: Let M be a surface into which G can be embedded. Without loss of generality, assume this embedding is cellular and so M has the same number of vertices and edges as

G. The **Euler characteristic** of a surface M, denoted $\chi(M)$, is calculated by the formula: $\chi(M) = V(M) - E(M) + F(M)$, where $V(M)$ is the number of vertices in M, $E(M)$ is the number of edges in M, and $F(M)$ is the number of faces in M. We know that $g(M) = \frac{2-\chi(M)}{2}$. We also know $F(M) \geq 1$ for any surface. By substitution: $g_{MAX}(G) = g(M) = \frac{2-\chi(M)}{2} = \frac{2-[V(M)-E(M)+F(M)]}{2} \leq \lfloor \frac{2-[V(M)-E(M)+1]}{2} \rfloor = \lfloor \frac{2-V(M)+E(M)-1}{2} \rfloor = \lfloor \frac{E(M)-V(M)+1}{2} \rfloor = \lfloor \frac{e-v+1}{2} \rfloor$.

Theorem 4.2.0.2 [1] *Let G be a connected graph with v vertices and e edges. Then,*

$$g_{MAX}(G) = \frac{1}{2}(e - v + 1) - \frac{1}{2} \min_{T} c_{\text{odd}}(G - E(T)),$$

where the minimum, if taken over all spanning trees T of G and $c_{\text{odd}}(G - E(T))$, denotes the number of components of $G - E(T)$ with an odd number of edges, and where $E(T)$ denotes the edges in tree T.

The last result we will add is the following:

Theorem 4.2.0.3 [1] *There exists a polynomially-bounded algorithm for finding the maximum genus of a connected graph.*

Classically, in the genus problem, a graph is defined first, and then one can draw a diagram from the vertices and edges given; however, sometimes it is more convenient to begin with the diagram and interpret this diagram as a graph. As we have mentioned before: if we begin with a diagram, we know that this is one diagram of many possible diagrams, and so finding the genus of our diagram will give an upper bound on the minimal genus taken over all diagrams; see Figure 4.3.

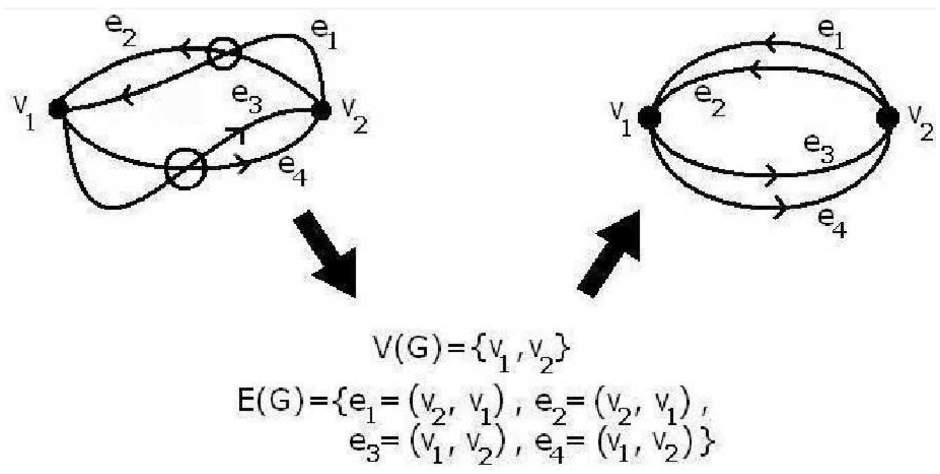


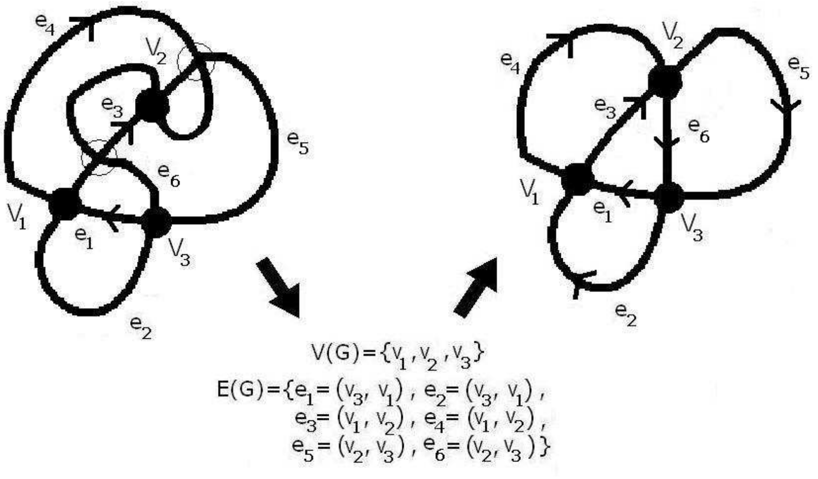
Figure 4.3: Beginning with a diagram, writing down the vertices and edges of the graph, and then drawing a new diagram for the graph shows that we can have two different surface embeddings related to the same graph. We will always indicate, usually by drawing circles, where interior points of edges intersect, which means in the diagram these points are not vertices. When the graph diagram is drawn on a surface with a high enough genus, these interior point intersections vanish. For a nonplanar graph, it is not possible to draw a planar diagram of the graph without these crossings.

4.3 The Genus of a Carter Surface

We have already used the Euler characteristic to calculate an upper bound for the genus of a graph. We will use the Euler characteristic of the Carter surface M to calculate its genus. Earlier we constructed the Carter surface of a Gauss code, a Gauss paragraph, and an LOC. All of the planar diagrams associated with these combinatorial objects can be interpreted as 4-regular multigraph diagrams, which allows us to apply the construct of the Carter surface to a 4-regular graph diagram. If we are given a 4-regular graph diagram with n vertices, $V(M) = n$, then the 4-regular property gives us the number of edges, $E(M) = 2n$. We place faces according to the construction of the Carter surface and count the number of faces, say

c faces, $F(M) = c$. After we have done this, then we can calculate the Euler characteristic of the Carter surface, $\chi(M)$. Using the following equation for orientable surfaces, we can know the genus of the Carter surface associated with the given 4-regular diagram: $g(M) = \frac{2-\chi(M)}{2}$, where $g(M)$ is the genus of M [1].

We want to emphasize that the genus of the Carter surface may not be the genus of the graph. Since we create the Carter surface from a given graph diagram, the Carter surface is dependent on the way the diagram is drawn. It is clear from Figure 4.3 that if we begin with a graph, as in the classical genus problem, then certain diagrams are more suited for the calculation of the genus than others. We have seen an altered version of the classical trefoil knot diagram in Figure 3.6 and we can easily calculate that the Carter surface has a genus of one, which means the diagram can be embedded in the torus. We could interpret the same graph using a different diagram and see that the graph is planar; see Figure 4.4.



$V(G)=\{v_1, v_2, v_3\}$
$E(G)=\{e_1=(v_3, v_1),\ e_2=(v_3, v_1),$
$e_3=(v_1, v_2),\ e_4=(v_1, v_2),$
$e_5=(v_2, v_3),\ e_6=(v_2, v_3)\}$

Figure 4.4: If we are given a diagram, we write the vertices and edges of the given graph diagram, and we draw a new diagram of the graph, then we can see that the Carter surface is not of the same genus as the genus of the graph. The diagram on the left has a Carter surface with genus one and the diagram on the right has a Carter surface of genus zero.

4.3.1 The Carter Surface Generalized to Any Graph Diagram

We have just seen that we can use the Carter surface as an upper bound for a 4-regular graph. Now consider the case where a vertex has a valency of five or more (i.e., we would like to extend the definition of Carter Surface to include all graph diagrams, not simply graph diagrams where each vertex has valency four).

Let Γ be a connected multigraph. The concept of "turn left" is still well defined because the diagram is projected onto a sphere; see Figure 4.5.

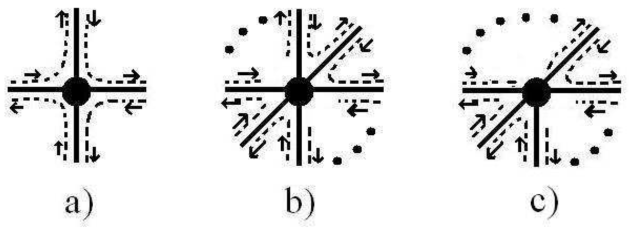


Figure 4.5: a) When a vertex has valency of four, we described the faces glued in by choosing edges in order by turning left. b) When a vertex has an even valency of six or more, we still have a well-defined turn left. c) When a vertex has an odd number of edges, there is still a well-defined turn left.

So to define the Carter surface, we need to begin at a vertex and follow an edge, turn left, and repeat this process at every vertex until we return to the starting point, defining a cycle. Since there are a finite number of edges, we have a finite number of left turns. That means, we will always be able to write down a cycle based on left turns. If we continue until all edges gave been used once in a positive direction and once in a negative direction, then we will have defined the Carter surface for which we can embed the multigraph. Since this diagram is one of many diagrams of the graph G associated with this diagram, we know that the genus of this surface is an upper bound for the genus of G.

4.4 Genus of Carter Surface in Linear Time

We have already seen that the genus of the Carter surface M associated with a given diagram is computed by $g(M) = \frac{2-\chi(M)}{2}$. Using the definition of Euler characteristic and construction of the Carter surface: $g(M) = \frac{2-(v-e+f)}{2}$, where v is the number of vertices and e is the number of edges in the given diagram, and f is the number of faces glued in by constructing M. Since the genus calculated by a sum and the only variable is f, which can be computed in linear time, we know that the genus of M is calculated in linear time [5], [7].

4.5 How Good Is the Carter Surface as an Upper Bound?

Note that by the construction of the Carter surface, the embedded graph diagram is cellular. When we remove the vertices and edges, the disks that were glued into the cell complex remain. That means that the graph embedded in its Carter surface is a cellular embedding of G [7].

Since the embedding of a graph diagram of G into the Carter surface is cellular, we can compare the genus of the Carter surface with $g_{MAX}(G)$. It is easy to see that the genus of the Carter surface of a given graph diagram yields an upper bound to the genus of a graph. A natural question when discussing upper bounds is: How "good" is this upper bound? Rephrasing the question: How close to the actual genus is the upper bound and how much of an improvement is this new upper bound in comparison to other upper bounds? We would also like to know when the diagram we start with is a ridiculous choice for a diagram, as in Figure 4.3, and when it is closer to the diagram yielding the minimal genus.

It is easy to see how this upper bound can be useful. It will never be "worse" than the available upper bounds, but sometimes it is much better. In Figure 4.6, we see a graph that fits the above requirements. The upper bounds that already exist give a genus upper bound of five, but by our method using the Carter surface, we calculate a genus upper bound to be one.

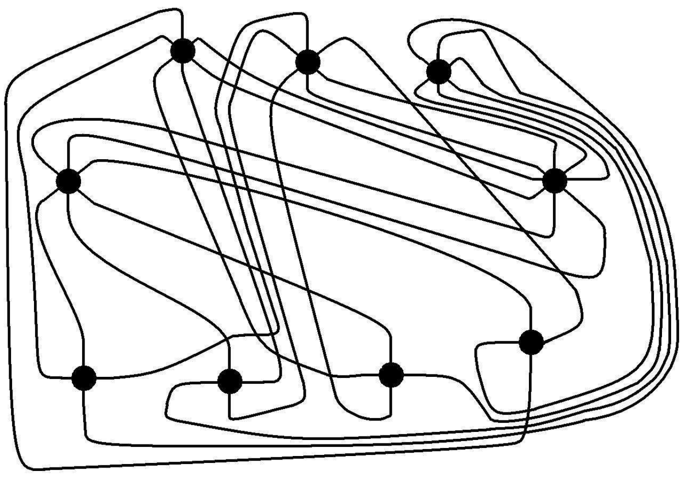


Figure 4.6: The given diagram has a Carter surface of genus one.

Consider the graph given in Figure 4.1. When we apply Theorem 4.2.0.1 to this example, the upper bound of the genus is four.

Using our method on this example, we have different diagrams we could use. If we use the diagram in Figure 4.1, then we produce a Carter surface of genus three. So the upper bound of the graph using the Carter surface is three. However, if we use the diagram in Figure 4.7, then we produce a Carter surface that has eight vertices, sixteen edges, and eight faces; see Figure 4.8. The Euler characteristic is zero, which means the genus of the Carter surface is one; see Figure 2.3. That means the genus of this particular graph is one, since we have already seen that this is a nonplanar graph and the upper bound is one. It is clear once again that the Carter surface is dependent on the diagram, therefore it is only a better upper bound if the diagram is chosen well. We could have chosen a diagram such that the Carter surface was of genus four and then our upper bound would not have been any more accurate than the ones already available.

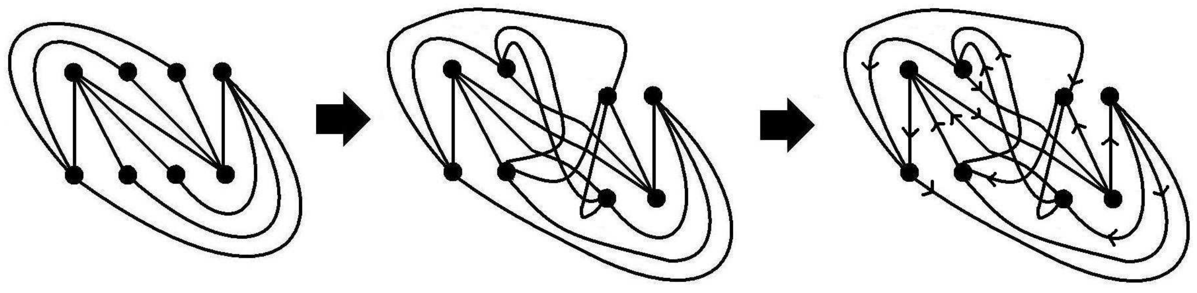


Figure 4.7: Place the nonplanar edges and then orient them so that the walk is always straight through at every vertex.

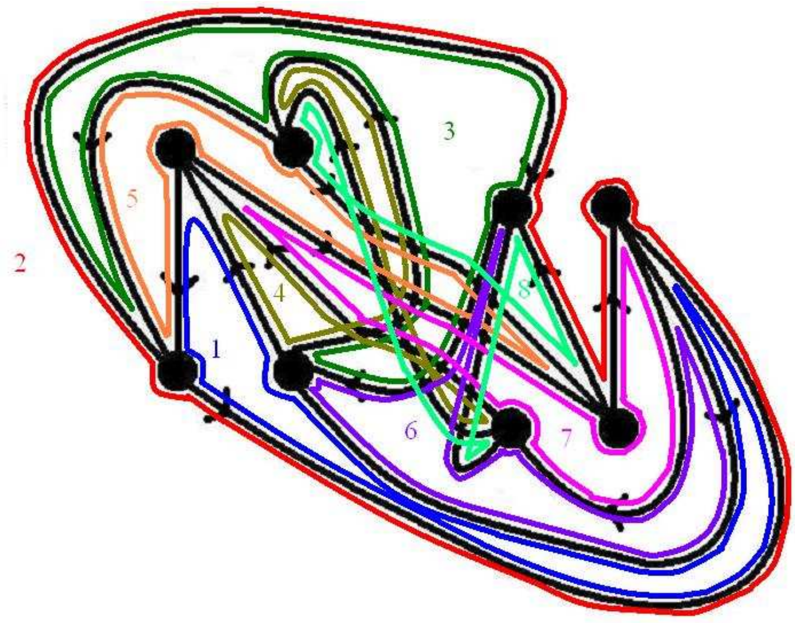


Figure 4.8: The different colors show that by using the turn left rule there will be eight different faces glued to produce the Carter surface.

CHAPTER 5

ALTERING A GIVEN DIAGRAM IN A NEIGHBORHOOD

A Carter surface is the minimal genus surface into which we can embed a given 4-regular graph diagram. Sometimes we have chosen a Euler walk on our diagram that consists of more than one circuit. Later in Section 6 of this paper, we discuss why it is beneficial to find an algorithm that changes the original diagram to one with a Euler walk consisting of a single circuit (or component) while at the same time the two diagrams have Carter surfaces with the same genus. Assuming the graphs are connected, it is clear that if we have two circuits, then they must share at least two vertices, call them a and b. Take a small neighborhood around one of the vertices, say a, cut the two edges in the neighborhood that start at a and eventually lead to b. The remainder is four "open-ended" edges, two connected to a and two connecting to the diagram outside of the neighborhood. Place a new vertex c in the neighborhood and connect the four "open-ended" edges to c; see Figure 5.1. We will have a finite number of circuits, so we can repeat this process a finite number of times until we have one circuit.

We claim that the genus of the Carter surface for the new diagram is the same as the genus of the Carter surface of the original diagram. We will use the Euler characteristic to prove our claim. Let the original diagram have v vertices, e edges, and f faces. The Euler characteristic of the original diagram's Carter surface is $v - e + f$. The new diagram is the same as the old everywhere except in the neighborhood, and so the Carter surfaces will be the same everywhere except in the neighborhood. When we look inside the neighborhoods and compare the change in vertices, edges, and faces, we see that we added one to v, two to e, and one to f; see Figure 5.2. The new Euler characteristic is calculated by $v+1-(e+2)+f+1 = v+1-e-2+f+1 = v-e+f$, which is the original Euler characteristic. That means the Carter surfaces have the same genus.

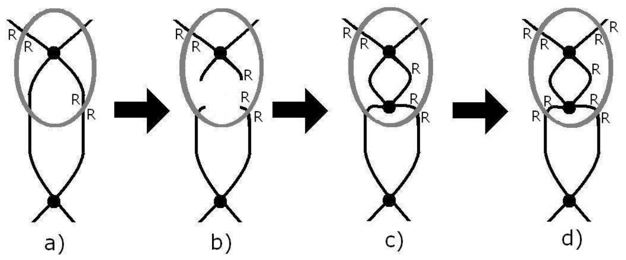


Figure 5.1: The "R" represents a strand we color red to distinguish one circuit from the other. How to alter the given diagram: a) Inside the neighborhood (gray), the surface is the same as the surface of a sphere; outside the neighborhood, the Carter surface has any finite number of handles. b) Cut the two edges so that there are four "open-ended" edges. c) Insert a new vertex and connect it to the four "open-ended" edges. d) The "R" labels are forced to the second circuit, so now we have one single circuit.

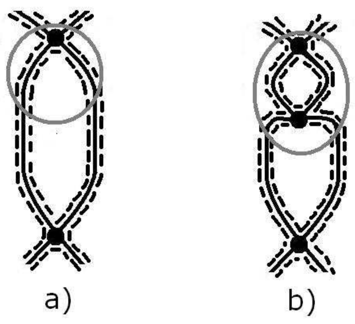


Figure 5.2: a) The dotted lines represent the way in which faces are glued. Inside the neighborhood, the dotted lines represent the exact way in which faces will be glued; however, outside the neighborhood, the dotted lines represent general patterns that will be followed when gluing in edges. b) Inside the neighborhood, the dotted lines represent the exact way in which faces will be glued according to the placement of the new vertex and edges. Outside the neighborhood, the dotted lines represent how the new diagram has not changed from the old diagram, so the general pattern that will be followed when gluing in edges remains the same. It is clear in the new diagram that we have an additional vertex, two additional edges, and one additional face.

CHAPTER 6

A PLANARITY TEST FOR GAUSS CODES

In [5], Cairns and Elton give a simple planarity test for Gauss codes. They compute certain numbers α_i and β_{ij} from the code and show that the Carter surface is a sphere if and only if all these invariants vanish. In this section, we review the work of Cairns and Elton [5], [6].

6.1 Computations from the Gauss Code $\alpha_i(w)$ and $\beta_{ij}(w)$

Let a Gauss word w with $2n$ letters be given. These two calculations, $\alpha_i(w)$ and $\beta_{ij}(w)$, provide an algorithm to read off of w whether or not the virtual knot associated with w is planar. First we must define certain sets: S_i is the subword of w between i^+ and i^-, not including these letters, $\bar{S}_i = i^+S_ii^-$, and S_i^{-1} is S_i only all the superscripts have reversed signs. Let the superscript $+ = 1$ and the superscript $- = -1$. For $i \in \{1, ..., n\}$, $\alpha_i(w)$ is the sum of the superscripts of all the letters in S_i. For $i, j \in \{1, ..., n\}$, $\beta_{ij}(w)$ is the sum of the superscripts of the of all the subwords that appear in both $\bar{S}_i$ and S_j^{-1}. We will use these calculations in the theorem later. For an example showing how to compute α_i's and β_{ij}'s for a given Gauss code, see Section 6.4.

6.2 Definition Alexander Numbering

An **Alexander numbering** of an oriented curve γ is a numbering of the faces of the Carter surface so that when traveling in the positive direction of γ the number of the face to the immediate right is always one less than the number of the face to the immediate left of γ

[5]. See Figure 6.1 for an example of a planar diagram along with a possible Alexander numbering.

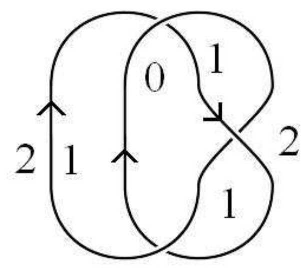


Figure 6.1: An example of an Alexander numbering for the classical trefoil knot.

6.3 Theorem 1 from Grant Cairns and Daniel M. Elton's Paper

We want to prove that the α_i's and β_{ij}'s test for planarity of a curve.

Theorem 6.3.0.1 [5] *Let w be an abstract Gauss word with k crossings. Then, w is the Gauss word of a closed normal planar curve if and only if $\alpha_i(w) = 0$ and $\beta_{ij}(w) = 0$ for all $i, j \in \{1, ..., k\}$.*

Proof of Theorem 6.3.0.1: Let w be an abstract Gauss word with k crossings and let γ be the associated normal curve on the Carter surface M_w. For each $i \in \{1, ..., k\}$, let the closed curve γ_i in M_w be the piece of γ starting at the crossing a_i^+ and finishing at a_i^-. That means that from the word w, the subword $\bar{S}_i$ of w is the subword that determines a restriction on γ.

Lemma 6.3.0.2 [5] *The Gauss word w is the Gauss word of a normal planar curve if and only if γ,γ_1,...,γ_k are all null-homologous.*

Proof of Lemma 6.3.0.2: If w is a Gauss word of a normal planar curve, then M_w has genus zero, and so all closed curves, specifically γ,γ_1,...,γ_k are null-homologous.

Assume γ,γ_1,...,γ_k, are null-homologous. Show that M_w has genus zero. We will show that M_w has a trivial first homology. Let c be a closed cycle on the polyhedral surface M_w (i.e. c is a closed chain formed by segments of γ). We will show that c is null-homologous, which will show that M_w has a trivial first homology.

For each $i \in \{1, ..., k\}$, define γ_i' to be the closed curve from a_i^- to a_i^+, which is $\gamma_i' = \gamma - \gamma_i$ including a_i and a_i^{-1}. Consider the vertices of M_w at which c "changes direction" by turning left at the vertex or turning right; see Figure 6.2. At such a vertex a_i, one may add a copy of γ_i, $-\gamma_i$, γ_i', or $-\gamma_i'$. Proceeding with this process for all of c, we travel all of the curve (and some), by only going straight (negatively or positively) on γ, so we have traveled an integer multiple of γ. So there exist integers r_i, s_i for $i = 1, ..., k$ and an integer t, such that

$$c + \sum_{i=1}^{k} r_i\gamma_i + \sum_{i=1}^{k} s_i\gamma_i' = t\gamma.$$

Since γ,γ_1,...,γ_k are all null-homologous, that means γ_1',...,γ_k' are null-homologous (since for each i, γ_i' is simply γ_i with the reverse orientation); and so, since c can be written as the sum of null-homologous closed curves, c must be null-homologous.

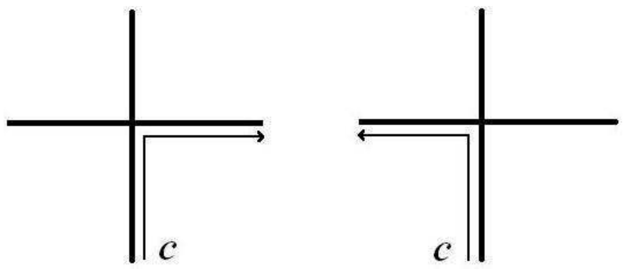


Figure 6.2: The curve turns to the left or the right at a vertex.

To use Lemma 6.3.0.2, and thus finish proving our theorem, we need to show that γ,γ_1,...,γ_k are null-homologous if and only if $\alpha_i(w) = 0$ and $\beta_{ij}(w) = 0$ for all $i, j \in \{1, ..., k\}$. First we will show that γ is null-homologous if and only if $\alpha_i(w) = 0$ for all $i \in \{1, ..., k\}$.

Lemma 6.3.0.3 [5] *γ is null-homologous if and only if γ has an Alexander numbering.*

Proof of Lemma 6.3.0.3: Let the faces of M_w be numbered $f_1, ..., f_m$. By definition, γ is null-homologous if and only if there are integers $a_1, ..., a_m$, such that $\gamma = \partial(a_1 f_1 + ... + a_m f_m)$. If γ is null-homologous, then $a_1, ..., a_m$ provide an Alexander numbering. If γ has an Alexander numbering, then this numbering forces every edge e_i to have a number one higher on the face to the left of e_i; let this face be numbered $k+1$, then on the face to the right of e_i, let this face be numbered k. When we use the Alexander numbering to provide the $a_1, ..., a_m$ (where a_i is on face f_i) for $\partial(a_1 f_1 + ... + a_m f_m)$, then we see that e_i is multiplied by $k+1$ in the positive direction and is multiplied by k in the negative direction; see Figure 6.3. Therefore, in $\partial(a_1 f_1 + ... + a_m f_m)$, we have $(k+1)e_i + k(-e_i)$, which reduces to exactly one e_i. This happens for all edges, and since we have exactly one of each edge in γ remaining from $\partial(a_1 f_1 + ... + a_m f_m)$, γ is null-homologous.

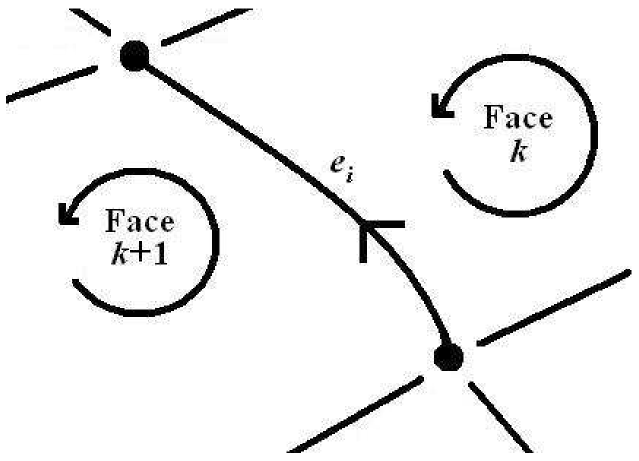


Figure 6.3: The given Alexander numbering provides the $a_1, ..., a_m$ for $\partial(a_1 f_1 + ... + a_m f_m)$ and so γ is null-homologous. Specifically, every e_i is found exactly once after calculating $\partial(a_1 f_1 + ... + a_m f_m)$.

Lemma 6.3.0.4 [5] *γ has an Alexander numbering if and only if $\alpha_i(w) = 0$ for all $i \in \{1, ..., k\}$.*

Proof of Lemma 6.3.0.4: We will prove the lemma by construction. For each face f, choose an interior point p_f as a vertex and place edges from p_f to the vertices of f, choosing to place the edges so they are mutually disjoint other than at p_f. We have divided f into a number of triangular regions, and each triangular region has a piece of γ as a base and p_f as a vertex (allowing for degenerate triangles for faces with one triangular region). For each non-vertex point of γ, there is a well-defined triangular region "to the left of γ" and "to the right of γ." Start at some non-vertex point on γ and call this point x. We will number the triangles to the left and right of γ in the following way: begin with the left triangle labeled with a one and the right triangle labeled with a zero. Travel along γ in the positive direction. We will label triangles to the left and right by the following rule: when we pass through a left-hand rule positive crossing, we will increase the subsequent labels for the left and right triangular regions by one and if we pass through a negative crossing, then we will decrease the subsequent labels for the left and right triangular regions by one. Once we return to the edge of γ that contains x, all the triangular regions will be labeled.

Claim: γ has an Alexander numbering if and only if for each face in M_w all triangular regions have the same numbering. If all triangular regions have the same numbering, then this numbering provides an Alexander numbering. Assume γ has an Alexander numbering. If we number the faces according to the Alexander numbering, then divide the faces into triangular regions, then all the triangular regions in the same face will have the same numbering, and by construction, these numbers coincide with the above procedure's labels (up to the addition of a constant).

Claim: All triangular regions in the same face of M_w have the same numbering if and only if $\alpha_i(w) = 0$ for all $i \in \{1, ..., k\}$. If $\alpha_i(w) = 0$ for all $i \in \{1, ..., k\}$, then there are two possible cases. The first case is that there is only one triangle in some of the faces, and thus no vertices to increase or decrease a neighboring triangle's number assignment, which also means all the triangular regions of that face agree in number since there is only one triangular region. The second case is that there are faces with two or more triangular regions. Let a_i be a positive vertex on such a face. If a_i is a positive vertex of a face, then a piece of γ_i is an edge of that face. Since $\alpha_i(w) = 0$, for every vertex on γ_i that causes an increase on the triangular region's number assignment, there is another vertex on γ_i that decreases the number assignment, thus when γ_i ends at a_i^{-1} the triangular regions are back to being the same number as when they started, and if we do this for each vertex on this particular face, we will see that all triangular regions must agree in the same face. Now assume all triangular regions in the same face have the same numbering. Beginning with one vertex on a face, if we travel along on edge, which is a section of γ, and come to another vertex we will either increase or decrease the numberings and continue to travel along another section of γ. If all the triangular regions have the same numbering, then the vertices between the first edge and the second edge must cause the same amount of increases and decreases; this is exactly what $\alpha_i(w) = 0$ calculates and if this happens for every face, then we have $\alpha_i(w) = 0$ for all $i \in \{1, ..., k\}$.

So now we have shown that γ is null-homologous if and only if $\alpha_i(w) = 0$ for all $i \in \{1, ..., k\}$.

To finish the proof, we need to show that γ_1,...,γ_k, are null-homologous if and only if $\beta_{ij}(w) = 0$ for all $i, j \in \{1, ..., k\}$.

Lemma 6.3.0.5 [5] *If γ is null-homologous, then γ_1,...,γ_k are null-homologous if and only if $\beta_{ij}(w) = 0$ for all $i, j \in \{1, ..., k\}$.*

Proof of Lemma 6.3.0.5: We will prove by a construction similar to the above procedure. First we want to fix i, and thus we fix a_i, a_i^{-1}, and γ_i. We will show that γ_i is null-homologous if and only if, when traveling on γ_i, the number of the face to the immediate right will always be one less that the number of the face to the immediate left, but, when traveling on γ_i', the number of the face to the immediate left will be the same as the number of the face to the immediate right (if this happens then the isolated curve γ_i has an Alexander numbering).

Similar to above, create triangular regions for each face. Proceed numbering triangles using the following rules: start at a point on γ_i, call it x, number the triangle to the immediate right zero and the triangle to the immediate left one. Travel along γ_i until you reach a vertex. For each vertex on γ_i that is not a_i or a_i^{-1}, do not increase or decrease the numberings on the triangular regions. At a_i, increase the number for the triangular region to the left by one, but do not change the number on the triangular region to the right. At a_i^{-1}, decrease the number for the triangular region to the left by one, but do not change the number on the triangular region to the right. When passing vertices on γ_i', use the following: for left-hand rule positive crossings, increase the number on the triangular regions to the immediate left and right by one; and for negative crossings, decrease the number on the triangular regions to the left and right by one. γ_i will be null-homologous if and only if in each face of M_w the triangles have the same numbering. Similar to the definition of $\alpha_i(w)$, $\beta_{ij}(w)$ calculates whether the triangles agree in number by traveling on γ_j and passing through the fixed γ_i. It is easy to see that γ_i is null-homologous if and only if $\beta_{ij}(w) = 0$ for all $i, j \in \{1, ..., k\}$.

6.4 Examples of α_i and β_{ij}

Do α_i and β_{ij} determine the genus of the Carter surface? In this section, we provide data that supports an affirmative answer of this question. We look at the first 40 examples of virtual knots from the list of Green and Bar-Natan [9], turn them into Gauss codes and compute the invariants. In all cases, the invariants distinguish virtual knots of different genus.

The following is an example of the calculations of α_i and β_{ij} for the Gauss Paragraph of the classical trefoil knot, which translates into the Gauss code: $\{1^+2^-3^+1^-2^+3^-\}$.

Table 6.1: Example of Computing S_i, $\overline{S}_i$, S_i^{-1}, and α_i.

i	S_i	$\overline{S}_i$	S_i^{-1}	α_i
1	2^-3^+	$1^+2^-3^+1^-$	2^+3^-	0
2	3^-1^+	$2^+3^-1^+2^-$	3^+1^-	0
3	1^-2^+	$3^+1^-2^+3^-$	1^+2^-	0

$\alpha_1 =$ the sum of the superscripts of $S_1 = -+ = 0$

$\alpha_2 =$ the sum of the superscripts of $S_2 = -+ = 0$

$\alpha_3 =$ the sum of the superscripts of $S_3 = -+ = 0$

So $\alpha_i = 0$ for all i.

Table 6.2: Example of Computing $\overline{S}_i \cap S_1^{-1}$, $\overline{S}_i \cap S_2^{-1}$, and $\overline{S}_i \cap S_3^{-1}$.

i	$\overline{S}_i \cap S_1^{-1}$	$\overline{S}_i \cap S_2^{-1}$	$\overline{S}_i \cap S_3^{-1}$
1	$1^+2^-3^+1^- \cap 2^+3^- = \emptyset$	$1^+2^-3^+1^- \cap 3^+1^- = 3^+1^-$	$1^+2^-3^+1^- \cap 1^+2^- = 1^+2^-$
2	$2^+3^-1^+2^- \cap 2^+3^- = 2^+3^-$	$2^+3^-1^+2^- \cap 3^+1^- = \emptyset$	$2^+3^-1^+2^- \cap 1^+2^- = 1^+2^-$
3	$3^+1^-2^+3^- \cap 2^+3^- = 2^+3^-$	$3^+1^-2^+3^- \cap 3^+1^- = 3^+1^-$	$3^+1^-2^+3^- \cap 1^+2^- = \emptyset$

So $\beta_{ij} = 0$ for $i, j = 1, 2, 3$.

The following are examples of different Gauss codes and their respective calculated α_i and β_{ij}. We also include the genus of the Carter surface associated with the Gauss code.

Table 6.3: Computing α_i and β_{ij} for Gauss code: $\{1^+2^+3^+1^-2^-3^-\}$, Genus = 1

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	2	0	1	2
2	0	-1	0	1
3	-2	-2	-1	0

Table 6.4: Computations for Gauss code: $\{1^+2^-3^+4^+2^+1^-4^-3^-\}$, Genus = 1

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	2	0	2	1	0
2	-2	-2	0	-1	0
3	0	-1	1	0	0
4	0	0	0	0	0

Table 6.5: Computations for Gauss code: $\{1^+2^-3^-4^+1^-4^-2^+3^+\}$, Genus = 2

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	-1	0	-2	-1	1
2	2	2	0	1	1
3	0	1	-1	0	1
4	-1	-1	-1	-1	0

Table 6.6: Computations for Gauss code: $\{1^+2^+1^-2^-3^+4^+3^-4^-\}$, Genus = 2

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	-1	-1	0	0	0
3	1	0	0	0	1
4	-1	0	0	-1	0

Table 6.7: Computations for Gauss code: $\{1^+3^-4^+1^-2^+4^-5^+2^-3^+5^-\}$, Genus = 2

i	α_i	$\beta_{i,1}$	β_{i2}	β_{i3}	β_{i4}	β_{i5}
1	0	0	1	-1	1	-1
2	0	-1	0	1	-1	1
3	0	1	-1	0	1	-1
4	0	-1	1	-1	0	1
5	0	1	-1	1	-1	0

Table 6.8: Computations for Gauss code: $\{1^+2^-3^-4^-5^+6^+1^-5^-2^+3^+4^+6^-\}$, Genus = 2

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}	β_{i5}	β_{i6}
1	-1	0	-2	-1	0	1	-1
2	2	2	0	1	2	0	0
3	0	1	-1	0	1	0	-1
4	-2	0	-2	-1	0	0	-2
5	0	-1	0	0	0	0	1
6	1	1	0	1	2	-1	0

Table 6.9: Computations for Gauss code: $\{1^+2^-3^-4^-5^+6^+1^-5^-2^+3^+4^+6^-\}$, Genus = 2

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}	β_{i5}	β_{i6}	β_{i7}	β_{i8}
1	0	0	0	-1	1	0	0	0	0
2	0	0	0	1	-1	0	0	0	0
3	0	1	-1	0	0	0	0	0	0
4	0	-1	1	0	0	0	0	0	0
5	0	0	0	0	0	0	0	-1	1
6	0	0	0	0	0	0	0	1	-1
7	0	0	0	0	0	1	-1	0	0
8	0	0	0	0	0	-1	1	0	0

According to the follow-up research paper by Cairns and Elton, we are able to write the tables with integers mod 2. If there are any non-zero computations mod 2, then the Gauss code does not yield a planar curve [6].

There is a table of virtual knots, classifying them up to four real crossings, by Jeremy Green. We will compute α_i and β_{ij} for the virtual knots 2.1 to 4.32 on that table [9]. We will label the tables in the following way: (Virtual knot number, Gauss code, Genus).

(2.1, $\{1^+2^+1^-2^-\}$, 1)

i	α_i	β_{i1}	β_{i2}
1	1	0	1
2	1	1	0

(3.1, $\{1^+3^+2^+3^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	1	0	0	0
2	0	0	0	1
3	1	0	1	0

(3.2, $\{1^+3^-2^-3^+1^-2^+\}$, 1)

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	1	0	0	1
2	0	0	0	0
3	1	1	0	0

(3.3, $\{1^+3^+2^+3^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	1	0	0	0
2	0	0	0	1
3	1	0	1	0

(3.4, $\{1^+3^+2^+3^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	1	0	0	0
2	0	0	0	1
3	1	0	1	0

$(3.5,\ \{1^+2^+3^+1^-2^-3^-\},\ 1)$

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	0	0	1	0
2	0	1	0	1
3	0	0	1	0

$(3.6,\ \{1^+2^-3^+1^-2^+3^-\},\ 0)$

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0

$(3.7,\ \{1^+2^-3^-1^-2^+3^+\},\ 1)$

i	α_i	β_{i1}	β_{i2}	β_{i3}
1	0	0	0	1
2	0	0	0	1
3	0	1	1	0

$(4.1,\ \{1^+2^+3^-4^-3^+4^+1^-2^-\},\ 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	1	1	0	0	0
3	1	0	0	0	1
4	1	0	0	1	0

$(4.2,\ \{1^+2^+3^+4^+3^-4^-1^-2^-\},\ 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	1	1
3	1	1	1	0	1
4	1	1	1	1	0

$(4.3, \{1^+2^+3^-4^+3^+4^-1^-2^-\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	1
2	1	1	0	0	1
3	1	0	0	0	1
4	1	1	1	1	0

$(4.4, \{1^+2^+3^-4^-3^+4^+1^-2^-\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	1	1	0	0	0
3	1	0	0	0	1
4	1	0	0	1	0

$(4.5, \{1^+2^+3^+4^+3^-4^-1^-2^-\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	1	1
3	1	1	1	0	1
4	1	1	1	1	0

$(4.6, \{1^+2^+3^+4^-3^-4^+1^-2^-\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	0
2	1	1	0	1	0
3	1	1	1	0	1
4	1	0	0	1	0

(4.7, $\{1^+2^+3^+4^+3^-4^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	1	1
3	1	1	1	0	1
4	1	1	1	1	0

(4.8, $\{1^+2^+3^-4^-3^+4^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	1	1	0	0	0
3	1	0	0	0	1
4	1	0	0	1	0

(4.9, $\{1^+3^-2^+4^-3^+4^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	0
4	1	0	0	0	0

(4.10, $\{1^+3^-2^+4^+3^+4^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	1
2	0	0	0	1	0
3	0	0	1	0	1
4	1	1	0	1	0

$(4.11, \{1^+3^+2^+4^-3^-4^+1^-2^-\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	0
2	0	0	0	0	0
3	0	1	0	0	0
4	1	0	0	0	0

$(4.12, \{1^+3^+2^+4^+3^-4^-1^-2^-\}, 1)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	1
2	0	0	0	0	0
3	0	1	0	0	1
4	1	1	0	1	0

$(4.13, \{1^+3^-2^-4^+3^+4^-1^-2^+\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	1	1	1	1	0

$(4.14, \{1^+3^+2^-4^-3^-4^+1^-2^+\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	0
2	0	0	0	1	1
3	0	1	1	0	0
4	1	0	1	0	0

(4.15, $\{1^+3^-2^+4^+3^+4^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	1
2	0	0	0	1	0
3	0	0	1	0	1
4	1	1	0	1	0

(4.16, $\{1^+3^-2^+4^-3^+4^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	0
2	0	0	0	1	0
3	0	0	1	0	0
4	1	0	0	0	0

(4.17, $\{1^+3^+2^+4^-3^-4^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	0
2	0	0	0	0	0
3	0	1	0	0	0
4	1	0	0	0	0

(4.18, $\{1^+3^-2^-4^+3^+4^-1^-2^+\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	1
2	0	0	0	0	1
3	0	0	0	0	1
4	1	1	1	1	0

$(4.19, \{1^+3^-2^-4^-3^+4^+1^-2^+\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	0
2	0	0	0	0	1
3	0	0	0	0	0
4	1	0	1	0	0

$(4.20, \{1^+3^+2^-4^+3^-4^-1^-2^+\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	1
2	0	0	0	1	1
3	0	1	1	0	1
4	1	1	1	1	0

$(4.21, \{1^+3^+2^-4^-3^-4^+1^-2^+\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	0
2	0	0	0	1	1
3	0	1	1	0	0
4	1	0	1	0	0

$(4.22, \{1^+3^+4^+2^+3^-4^-1^-2^-\}, 2)$

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	0	1
3	0	1	0	0	1
4	0	1	1	1	0

(4.23, $\{1^+3^-4^-2^-3^+4^+1^-2^+\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	1	1	0	1	0
3	0	0	1	0	1
4	0	0	0	1	0

(4.24, $\{1^+3^+4^+2^-3^-4^-1^-2^+\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	1	0
3	0	1	1	0	1
4	0	1	0	1	0

(4.25, $\{1^+3^-4^-2^+4^+3^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	0	1
3	1	1	0	0	0
4	1	1	1	0	0

(4.26, $\{1^+3^+4^+2^+4^-3^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	1	1	0	0	1
3	1	0	0	0	0
4	1	0	1	0	0

(4.27, $\{1^+3^-4^-2^-4^+3^+1^-2^+\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	1	1
2	1	1	0	0	1
3	1	1	0	0	0
4	1	1	1	0	0

(4.28, $\{1^+3^+4^+2^-4^-3^-1^-2^+\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	1	0	0
2	1	1	0	0	1
3	1	0	0	0	0
4	1	0	1	0	0

(4.29, $\{1^+3^-4^-3^+2^+4^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	0
2	0	0	0	1	0
3	1	0	1	0	0
4	0	0	0	0	0

(4.30, $\{1^+3^-4^+3^+2^+4^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	0	1
2	0	0	0	1	1
3	1	0	1	0	0
4	0	1	1	0	0

(4.31, $\{1^+3^+4^-3^-2^+4^+1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	0
2	0	0	0	1	0
3	1	1	1	0	1
4	0	0	0	1	0

(4.32, $\{1^+3^+4^+3^-2^+4^-1^-2^-\}$, 2)

i	α_i	β_{i1}	β_{i2}	β_{i3}	β_{i4}
1	1	0	0	1	1
2	0	0	0	1	1
3	1	1	1	0	1
4	0	1	1	1	0

The following result summarizes our findings:

Theorem 6.4.0.6 *Let w_1 and w_2 be Gauss codes obtained from any of the virtual knots 2.1 to 4.32 on the Green/Bar-Natan virtual knot table. If w_1 and w_2 have the same α_i and β_{ij}, then they have the same genus.*

Whether we can compute the Euler characteristic of a Gauss code from α_i and β_{ij} is a topic for further research.

REFERENCES

[1] Bojan Mohar and Carsten Thomassen, *Graphs on Surfaces*, The Johns Hopkins University Press (2001).

[2] W.D. Wallis, *A Beginner's Guide to Graph Theory*, Birkhäuser (2000), 1-9.

[3] J. Harlander, S. Rosebrock, *On Distinguishing Virtual Knot Groups from Knot Groups*, to appear in J. Knot Theory Ramifications (2010).

[4] L.H. Kauffman, *Virtual Knot Theory*, Europ. J. Combinatorics (1999), **20**, 663-691.

[5] G. Cairns and D.M. Elton, *The Planarity Problem for Signed Gauss Words*, J. Knot Theory Ramifications (1993), **2**, 359-367.

[6] G. Cairns and D.M. Elton, *The Planarity Problem II*, J. Knot Theory Ramifications (1996), **5**, 137-144.

[7] V. Kurlin, *Gauss paragraphs of classical links and a characterization of virtual link groups*, Math. Proc. Camb. Soc. (2008), **145**, 129-140.

[8] J.S. Carter, *Classifying Immersed Curves*, Amer. Math. Soc. (1991), **111**, 281-287.

[9] Jeremy Green and Dror Bar-Natan, *A Table of Virtual Knots*. 10 August 2004, Web. 21 February 2010 ⟨http://www.math.toronto.edu/∼drorbn/Students/GreenJ/⟩.