2023

Surjectivity of the Wahl Map on Cubic Graphs

Angela C. Hanson
University of Kentucky, hansonac13@gmail.com
Author ORCID Identifier:
https://orcid.org/0000-0002-8852-9130
Digital Object Identifier: https://doi.org/10.13023/etd.2023.132

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Angela C. Hanson, Student

Dr. David Jensen, Major Professor

Dr. Benjamin Braun, Director of Graduate Studies

Surjectivity of the Wahl Map on Cubic Graphs

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Arts and Sciences
at the University of Kentucky

By
Angela C. Hanson
Lexington, Kentucky

Director: Dr. David Jensen, Professor of Mathematics
Lexington, Kentucky
2023

https://orcid.org/0000-0002-8852-9130

ABSTRACT OF DISSERTATION

Surjectivity of the Wahl Map on Cubic Graphs

Much of algebraic geometry is the study of curves. One tool we use to study curves is whether they can be embedded in a $K3$ surface or not. If the Wahl map is surjective on a curve, that curve cannot be embedded in a $K3$ surface. Therefore, studying if the Wahl map is surjective for a particular curve gives us more insight into the properties of that curve. We simplify this problem by converting graph curves to dual graphs. Then the information for graphs can be used to study the underlying curves. We will discuss conditions for the Wahl map to be surjective on a cubic graph. The Wahl map on a cubic graph has two parts, one map onto the vertices and another onto the edges. We found that if the cubic graph is 3-edge-connected and non-planar, then the map on the vertices is surjective. Surjectivity of the map on the edges is not as clear. However, if the Wahl map is surjective on a cubic graph, the girth of the graph must be at least 5. Finally, based on data collected, we have observed that as the size of cubic graphs of girth at least 5 increases, the probability that the Wahl map is surjective on those graphs appears to approach 1.

KEYWORDS: Wahl map, graphs, graph curves

Angela C. Hanson

April 6, 2023

Surjectivity of the Wahl Map on Cubic Graphs

By
Angela C. Hanson

Dr. David Jensen
Director of Dissertation

Dr. Benjamin Braun
Director of Graduate Studies

April 6, 2023
Date

ACKNOWLEDGMENTS

There are so many people who have influenced my journey to this point. I would like to take this opportunity to express my appreciation explicitly.

When I was in elementary school, one of my teachers, Mrs. Tracey Laubert, saw that I had a talent and passion for mathematics. She offered me an opportunity to learn middle-school-level math while the rest of the class was working on their math lessons. Mrs. Laubert was the first teacher to feed my mathematical curiosity and help me develop as a leader by sending me to a youth leadership conference. After these experiences in her class, I aspired to teach mathematics someday like she taught me. This is when I first considered going to graduate school and becoming a math professor. Then in middle school and high school, I had teachers like Mrs. Colleen Sutliff and Mr. David Sutliff who helped me take what I knew about math and expand on it. They each gave me opportunities to practice explaining mathematics so I could develop a deeper understanding and help my classmates at the same time.

By college, I knew that I wanted to study math, but I was just beginning to see what that really meant. Up until that point, my math classes were laid out in a particular order, but in college I was exposed to a whole buffet of math topics. Drs. Wayne Tarrant, Tom Langley, and Leanne Holder mentored me through this exploration. I started with a focus in statistics but later fell in love with combinatorics and graph theory. These three professors helped me navigate my options and find my passions. Ultimately, they were the ones that helped me decide to go to graduate school and supported me as I looked for a program that would be a good fit. I admire each of them as educators and mathematicians, but I also learned so much from them personally. They helped me find my way.

When I finally got to graduate school, I still had a lot to learn. My first-year professors, Drs. Peter Perry, Chris Manon, Uwe Nagel, and Ben Braun, helped me adjust to graduate-level mathematics and to living in a new city. They were each very understanding of all the big life changes we were facing. When I was confused with the material or overwhelmed by life, I could always count on them to support me and help me through it. In particular, as my faculty mentor, Ben has been a sounding board for me on matters ranging from progression in the program to everyday life. Last, but certainly not least, I want to thank my advisor, Dr. Dave Jensen; he has taught me so much mathematically and professionally in the last four years. When life threw me curve balls, he helped me prioritize my efforts. When I got stuck on a problem, he gave me a fresh perspective. When I needed additional support, he was the first to advocate for me. He has always been on my side and has guided me through this experience. I cannot thank him enough for the investment of his time, energy, and emotions.

While these educators played a significant role in my education and career, I could not have made it through this process without my family. I want to thank my parents and brother for supporting me throughout my life, but especially in graduate school. They have always made me feel loved and encouraged. When I reached major crossroads in my education and career, they have offered a fresh perspective and unconditional support. My dad, Jim, is a compassionate college professor. He has been a role model for me in that respect but never expected me to follow in his footsteps. My mom, Diane, has helped me learn to balance my personal and professional goals so that I can give my best in both areas. When I have reached my limits, she has also reminded me that it is okay to prioritize self-care and give myself grace. My brother, Keith, has helped me with the programming for my research when I got stuck, but more than that, he let me lean on him when I had a bad day. Both my parents and my brother have reminded me that I am enough, no matter what is

thrown my way.

Beyond my immediate family, I want to thank my partner's parents, Jeff and Jenney Holthouse and Jennie Bailey, for accepting me into their family and supporting me through graduate school. They have understood when I needed to study for preliminary exams during visits and when we couldn't visit at all because I had to work. When I have had big decisions to make, they have happily given me advice and guidance. They have doubled my family support, and for that, I could not be more grateful. Similarly, my extended family and closest family friends have cheered me on at each milestone and continue to encourage me as I face this next chapter of my life. I am eternally grateful for my entire family.

My friends in graduate school have also become my family. I started this program with some of my closest friends, Courtney George, Katie Bruegge, and Lewis Dominguez. We have been through classes and preliminary exams together. When I have needed them most, they have each been there for me, as I have tried to be there for them too. We have a special bond through mutual experience, but we have also chosen to feed into our friendships beyond the University of Kentucky. I hope to continue to call them friends as we go our separate ways and start our careers. Along the way, I have also leaned on other graduate students like Kalila Lehman, Deborah Blevins, Shane Clark, Julie Vega, Sara Church, and Austin Alderete. Each of them supported me personally and academically. For that, I will always be grateful.

Most of all, I want to thank my partner, Connor Holthouse. He as been my rock for over six years, and I couldn't have done this without him. He has held me on the bad days and has celebrated with me on the good days. No matter what we have faced, we have done so together. Connor is so generous, supportive, and brilliant. When I need advice or just need help completing a sentence, he is there for me. I cherish his partnership and value his input. I will never have the words to adequately communicate how much I love and appreciate Connor, but I will continue to try as

long as we are together.

There have been numerous people who have influenced my story and shaped who I am; I am grateful to them all. Ultimately though, I give the final credit to my God for placing these people in my life and loving me unconditionally.

CONTENTS

Acknowledgments iii

List of Tables viii

List of Figures ix

Chapter 1 Introduction 1
1.1 The Wahl Map $\mathbb{W}_C$ on a Curve 1
1.2 Primary Results 1

Chapter 2 The Wahl Map $\mathbb{W}_G$ on Cubic Graphs 4
2.1 Specialization to Graphs 4
2.2 Graphic Definition of the Wahl Map 5
2.3 Rank Independence 10

Chapter 3 Surjectivity of the Vertex Map $\mathbb{W}_G^0$ 16
3.1 Properties of Cubic Graphs 16
3.2 Surjectivity Criteria 18

Chapter 4 Surjectivity of the Overall Map $\mathbb{W}_G$ 20
4.1 Girth of Graphs 20
4.2 Programming Results 22

Appendices 25
Appendix A: SageMath Program 25

Bibliography 36

Curriculum Vita 37

LIST OF TABLES

4.1 A sampling of random cubic graphs with girth at least 5, of various sizes. 22

LIST OF FIGURES

2.1 An example of a graph curve C and its dual graph G. 4
2.2 The ascending orientation ω and (123)-orientation φ on $K_{3,3}$. 6
2.3 The graph cycle $c = (e_{12}, e_{24}, e_{34}, e_{13})$ in graph $K_{3,3}$. 7
2.4 A visual depiction of the notation in the definition of $\mathbb{W}^0_G$. 8
2.5 A visual depiction of the notation in the definition of $\mathbb{W}^1_G$. 8

3.1 Three disjoint paths forming a pair of graph cycles c_1, c_2 that overlap precisely on edge e. 16
3.2 The maps defined for Theorem 3.1.3. 17
3.3 Smoothing vertex w to form H''. 18
3.4 This is the vertex labeling of $K_{3,3}$ used in this proof. 19

4.1 The 3-cycle σ and its neighbors as labeled in the proof. 21
4.2 The cycles through T_3. 22
4.3 The linear combinations of τ_4, τ_5, τ_6 to form $\sigma, \tau_1, \tau_2, \tau_3$. 23
4.4 The Heawood graph. 24

Chapter 1 Introduction

1.1 The Wahl Map $\mathbb{W}_C$ on a Curve

One way that we study a curve C is by determining if C can be embedded in a $K3$ surface. Non-surjectivity of the Wahl map $\mathbb{W}$ is a necessary condition for an algebraic curve to be embedded in a $K3$ surface:

Theorem 1.1.1. *[Wah87] If a smooth algebraic curve C can be embedded in a $K3$ surface, then $\mathbb{W}$ cannot be surjective on C.*

Therefore, studying when the Wahl map is surjective helps us better understand curve embeddings in $K3$ surfaces and, consequently, the curves themselves.

The Wahl map is a special case of the Gauss map. For a line bundle L on a curve C and the canonical line bundle K on C, the Gauss map

$$\Phi_L : \bigwedge^2 H^0(C, L) \to H^0(C, K \otimes L^2)$$

is defined by $\Phi_L(\sigma \wedge \tau) = \sigma \otimes d\tau - \tau \otimes d\sigma$ and extending linearly. The Wahl map is $\mathbb{W} = \Phi_K$. Since the Wahl map depends on the choice of curve, we will use $\mathbb{W}_C$ to denote the Wahl map for a particular curve C.

One of the earliest results about surjectivity of the Wahl map comes from Ciliberto, Harris, and Miranda:

Theorem 1.1.2. *[CHM88] If C is a general curve of genus $g \geq 10$ and $g \neq 11$, then $\mathbb{W}_C$ is surjective.*

Notice that this result is only for *general* curves and not for all curves. The proof of Theorem 1.1.2 uses graph theory but only looks at a particular family of graphs that are a generalization of the Petersen graph. Our goal is to expand these results to a larger class of graphs.

1.2 Primary Results

We study when $\mathbb{W}_C$ is surjective by specializing to graphs, an approach motivated by the work of Ciliberto, Harris, and Miranda. For a graph curve C, we can convert C to a graph G, as described in Section 2.1. Then we define $\mathbb{W}_G$ and study when it is surjective to better understand when $\mathbb{W}_C$ is.

Let $\mathrm{Cat}_0(G, \mathbb{C})$ denote the $|V(G)|$-dimensional $\mathbb{C}$-vector space $\mathbb{C}^{V(G)}$ of 0-chains with coefficients in $\mathbb{C}$ on the graph G. An element of $\mathrm{Cat}_0(G, \mathbb{C})$ can be written as $\sum_{v \in V(G)} c_v v$ with $c_v \in \mathbb{C}$. Similarly, let $\mathrm{Cat}_1(G, \mathbb{C})$ denote the $|E(G)|$-dimensional

$\mathbb{C}$-vector space $\mathbb{C}^{E(G)}$ of 1-chains with coefficients in $\mathbb{C}$ on the graph G. An element of $\mathrm{Cat}_1(G,\mathbb{C})$ can be written as $\sum_{e\in E(G)} c_e e$ with $c_e \in \mathbb{C}$. The Wahl map on graphs

$$\mathbb{W}_G : \bigwedge^2 H_1(G,\mathbb{C}) \to \mathrm{Cat}_0(G,\mathbb{C}) \oplus \mathrm{Cat}_1(G,\mathbb{C})$$

takes in a linear combination of wedge products of cycles from a cubic graph G and returns a linear combination of the vertices and edges of G. We explicitly define this map in Section 2.2. It is worth noting that this definition depends on a choice of vertex and edge orientations which is not clear from the notation. However, we will show in Section 2.3 that the rank of $\mathbb{W}_G$ is independent of this choice.

Since the codomain of $\mathbb{W}_G$ is a direct sum, the map $\mathbb{W}_G$ decomposes as $\mathbb{W}_G = \mathbb{W}_G^0 \oplus \mathbb{W}_G^1$, where the codomain of $\mathbb{W}_G^0$ is generated by the vertices of the graph and the codomain of $\mathbb{W}_G^1$ is generated by the edges of the graph. The important take-away is that we can consider our problem in two pieces:

1. When is $\mathbb{W}_G^0$ surjective?
2. When is $\mathbb{W}_G|_{\ker \mathbb{W}_G^0}$ surjective?

In Chapter 3, we attempt to find necessary and sufficient graphic criteria to answer the first question. We have some conditions from Miranda.

Theorem 1.2.1. *[Mir89] Let G be a cubic, 3-edge-connected graph. If G is planar, then $\mathbb{W}_G^0$ is not surjective.*

In Section 3.1, we build the tools needed to prove the converse:

Theorem 1.2.2. *Let G be a cubic, 3-edge-connected graph. If G is non-planar, then $\mathbb{W}_G^0$ is surjective.*

In order to prove this result, we use that surjectivity of $\mathbb{W}_G^0$ is preserved under refinement:

Theorem 1.2.3. *Let G and H be cubic graphs. Suppose G is 3-edge-connected and H is a topological minor of G. If $\mathbb{W}_H^0$ is surjective, then $\mathbb{W}_G^0$ is also surjective.*

Note that when H is a topological minor of G, G contains a subgraph H' that is isomorphic to a refinement of H. For cubic graphs, containing a refinement of $K_{3,3}$ is equivalent to being non-planar. Thus by Theorem 1.2.3 and surjectivity of $\mathbb{W}_{K_{3,3}}^0$, we prove Theorem 1.2.2 in Section 3.2.

It remains to determine for which graphs G the map $\mathbb{W}_G$ restricted to $\ker(\mathbb{W}_G^0)$ is surjective. In Section 4.1, we discuss how this question relates to the girth of G.

Theorem 1.2.4. *If $\mathbb{W}_G$ is surjective, then* $\mathrm{girth}(G) \geq 5$.

The heart of the argument for Theorem 1.2.4 is that $\mathbb{W}_G$ is not locally surjective on 2-cycles, 3-cycles, or 4-cycles. Unfortunately, the converse of Theorem 1.2.4 is not true. For example, the Heawood graph H_{14} is a cubic, 3-edge-connected, non-planar

graph of girth 6, but $\mathbb{W}_{H_{14}}$ is not surjective. In Section 4.2, we provide experimental data suggesting that the Wahl map is surjective for many cubic graphs of girth at least 5, despite some counterexamples.

Conjecture 1.2.5. *Let F_g be the event that $\mathbb{W}_G$ is surjective for a random cubic graph G of genus g and* $\text{girth}(G) \geq 5$. *Then we have*

$$\lim_{g\to\infty} P(F_g) = 1.$$

Chapter 2 The Wahl Map $\mathbb{W}_G$ on Cubic Graphs

2.1 Specialization to Graphs

Like Ciliberto, Harris, and Miranda, we will examine the Wahl on graph curves [CHM88].

Definition 2.1.1. A **graph curve** is a connected curve C that is a union of projective lines satisfying the following conditions:

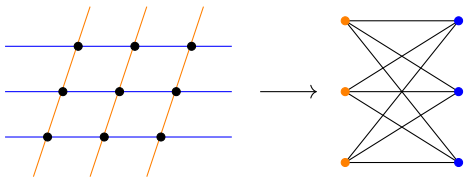
- Each line meets exactly three others,
- The intersection of two lines is a single point, and
- At most two lines intersect at a given point.

Graph curves can then be converted to graphs, which we will use in our study of the Wahl map.

Definition 2.1.2. For a graph curve C, the **dual graph** G of C is constructed in the following way:

- Each line in C corresponds to a vertex in G.
- If two lines in C intersect, there is an edge between the corresponding vertices in G.

Example 2.1.3. See Figure 2.1 for an example graph curve C and its dual graph G. The orange, vertical lines of C are parallel, so the corresponding vertices do not have edges between them in G. Similarly, the blue, horizontal lines of C are parallel, and the corresponding vertices do not have edges between them in G. For every point where an orange line intersects a blue line, we have a corresponding edge in G.



graph curve C dual graph G

Figure 2.1: An example of a graph curve C and its dual graph G.

Observation 2.1.4. In a graph curve, each line intersects three others, so the dual graph is always cubic.

We define the Wahl map on these dual graphs in the next section.

2.2 Graphic Definition of the Wahl Map

For any finite graph G, we label the n vertices of G with v_i for $i = 0, \ldots, n-1$. We fix such a vertex labeling for the remainder of our discussion. Let e_{ij} denote an edge between v_i and v_j, regardless of orientation. That is $e_{ij} = e_{ji}$.

In order to consider the Wahl map on a cubic graph G, we first need to define edge and vertex orientations on G.

Definition 2.2.1. For a graph G, an **edge orientation** ω is an assignment of a direction for each edge in G from one endpoint to another.

If an edge e_{ij} is directed from vertex v_i to vertex v_j in an edge orientation ω, we will denote this with $\omega(e_{ij}) = v_i \to v_j$.

Definition 2.2.2. Let ω direct the edges of G in ascending order according to vertex index, so for edge $e_{ij} = (v_i, v_j)$ we have $\omega(e_{ij}) = v_i \to v_j$ if $i < j$ and $\omega(e_{ij}) = v_j \to v_i$ otherwise. We call ω the **ascending orientation**.

Since the vertex labels are distinct and indexed with the natural numbers, they have a proper ordering, so the ascending orientation is an edge orientation.

Definition 2.2.3. For each vertex v in a graph G, choose a cyclic permutation φ_v of the edges adjacent to v. A **vertex orientation** φ is the collection of such φ_v over all vertices v in G.

For a cyclic permutation φ_v on the edges adjacent to vertex v, we will use $\varphi_v(e)$ to denote the edge that follows e in the permutation of edges adjacent to v.

Definition 2.2.4. Let G be a cubic graph and let vertex v_i have adjacent edges e_{ij}, e_{ik}, e_{im} where $j < k < m$. Define φ_{v_i} in the following way:

- $\varphi_{v_i}(e_{ij}) = e_{ik}$
- $\varphi_{v_i}(e_{ik}) = e_{im}$
- $\varphi_{v_i}(e_{im}) = e_{ij}$.

Then we call the set $\{\varphi_{v_i}\}_{i=0}^{n-1}$, defined this way, the **(123)-orientation**.

For all vertices v in G, (123) is a cyclic permutation on the edges adjacent to v. Hence the (123)-orientation is a vertex orientation.

Example 2.2.5. Consider an explicit example on $K_{3,3}$, as depicted in Figure 2.2, with the ascending and (123)-orientations. The arrows on each edge indicate the ascending orientation and the circular arrows to the side of each vertex indicate the (123)-orientation. For the neighbors of v_0, we have $\varphi_{v_0}(e_{01}) = e_{03}$, $\varphi_{v_0}(e_{03}) = e_{05}$, $\varphi_{v_0}(e_{05}) = e_{01}$.

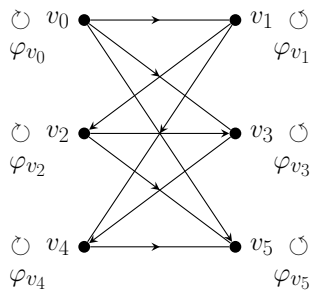


Figure 2.2: The ascending orientation ω and (123)-orientation φ on $K_{3,3}$.

Consider the boundary map

$$\partial : \mathrm{Cat}_1(G, \mathbb{C}) \to \mathrm{Cat}_0(G, \mathbb{C})$$

defined with the edge orientation ω by

$$\partial(e_{ij}) = \begin{cases} v_j - v_i & \text{if } \omega(e_{i,j}) = v_i \to v_j \\ v_i - v_j & \text{if } \omega(e_{i,j}) = v_j \to v_i \end{cases}.$$

We use $\partial_v(e)$ to denote the coefficient of v in $\partial(e)$. If e is oriented from v_1 to v_2, then $\partial(e) = v_2 - v_1$ so

$$\partial_v(e) = \begin{cases} -1 & \text{if } v = v_1 \\ 1 & \text{if } v = v_2 \\ 0 & \text{otherwise} \end{cases}.$$

Definition 2.2.6. An element $\sigma \in \mathrm{Cat}_1(G, \mathbb{C})$ is in $H_1(G, \mathbb{C})$ if and only if $\partial(\sigma) = 0$. We call $\sigma \in H_1(G, \mathbb{C})$ a **homology cycle**.

Example 2.2.7. Consider

$$\sigma = e_{12} + e_{24} - e_{34} - e_{13}$$

under the ascending orientation. Then we have

$$\partial(\sigma) = v_2 - v_1 + v_4 - v_2 - v_4 + v_3 - v_3 + v_1 = 0$$

so σ is a homology cycle. We use σ_e to denote the coefficient of e in σ, so for

$$\sigma = e_{12} + e_{24} - e_{34} - e_{13}$$

we have $\sigma_{e_{12}} = 1$, $\sigma_{e_{34}} = -1$, and $\sigma_{e_{14}} = 0$.

Definition 2.2.8. Consider a sequence c of edges that connect a sequence of vertices in a graph G. We call c a **graph cycle** if the following are satisfied:

- Each edge in c is distinct, and

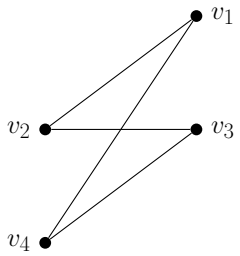


Figure 2.3: The graph cycle $c = (e_{12}, e_{24}, e_{34}, e_{13})$ in graph $K_{3,3}$.

- Only the first and last vertices of c are equal.

Example 2.2.9. Consider

$$c = (e_{12}, e_{24}, e_{34}, e_{13})$$

connecting vertices

$$v_1, v_2, v_4, v_3, v_1.$$

No edge repeats in this sequence, and vertex v_1 is where the sequence starts and ends. Then c is a graph cycle, as shown in Figure 2.3.

Given the edges of a graph cycle c in graph G, we can form a linear combination $\sigma_c \in H_1(G, \mathbb{C})$ of those edges in the following way. The coefficient of $e_{ij} \in c$ in the linear combination σ_c is determined by edge orientation ω so that $\partial(\sigma_c) = 0$. Therefore, once an edge orientation ω is fixed, we can form homology cycles from graph cycles.

Now we provide a definition for the Wahl map on a cubic graph G with edge and vertex orientations ω, φ.

Definition 2.2.10. [CF92] Let G be a cubic graph with edge orientation ω and vertex orientation φ. Let v be a vertex of G that is adjacent to edges e_1, e_2, e_3. Without loss of generality, index the edges so that $\varphi_v(e_1) = e_2$, $\varphi_v(e_2) = e_3$, and $\varphi_v(e_3) = e_1$ (See Figure 2.4). Then we define a map on the vertices

$$\mathbb{W}_G^0 : \bigwedge^2 H_1(G, \mathbb{C}) \to \mathrm{Cat}_0(G, \mathbb{C})$$

by

$$\mathbb{W}_G^0(\sigma \wedge \tau)_v = \partial_v(e_1)\partial_v(e_2)\det \begin{pmatrix} \sigma_{e_1} & \tau_{e_1} \\ \sigma_{e_2} & \tau_{e_2} \end{pmatrix}$$

and extending linearly.

This map effectively takes pairs of homology cycles of G and returns values on the vertices of G. There is a similar map that effectively takes pairs of cycles in G and returns values on the edges of G.

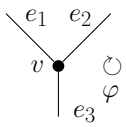


Figure 2.4: A visual depiction of the notation in the definition of $\mathbb{W}_G^0$.

Definition 2.2.11. [CF92] For a cubic graph G with edge orientation ω and vertex orientation φ, let e be an edge of G that is adjacent to vertices v_1, v_2. Label the remaining edges adjacent to v_1 with e_1, e_3 and the remaining edges adjacent to v_2 with e_2, e_4, where we have indexed the edges so that $\varphi_{v_1}(e) = e_1$, $\varphi_{v_1}(e_1) = e_3$, $\varphi_{v_2}(e) = e_2$, and $\varphi_{v_2}(e_2) = e_4$ (See Figure 2.5). Then we define a map on the edges

$$\mathbb{W}_G^1 : \bigwedge^2 H_1(G, \mathbb{C}) \to \mathrm{Cat}_1(G, \mathbb{C})$$

by

$$\mathbb{W}_G^1(\sigma \wedge \tau)_e = \partial_{v_1}(e_1)\partial_{v_2}(e_2)\det \begin{pmatrix} \sigma_{e_1} & \tau_{e_1} \\ \sigma_{e_2} & \tau_{e_2} \end{pmatrix}$$

and extending linearly.

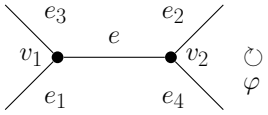


Figure 2.5: A visual depiction of the notation in the definition of $\mathbb{W}_G^1$.

Note that we abuse notation in Definitions 2.2.10 and 2.2.11. The map $\mathbb{W}_G^0$ depends on a choice of which edge is labeled e_1, vertex orientation φ, and edge orientation ω. The map $\mathbb{W}_G^1$ depends on a choice of vertex and edge orientations φ, ω. However, we will show in Section 2.3 that $\mathrm{rk}(\mathbb{W}_G)$ is independent of these choices.

Example 2.2.12. Let's continue with our earlier example on $K_{3,3}$, Example 2.2.5, with the ascending edge orientation and (123)-vertex orientation. For $\mathbb{W}_{K_{3,3}}^0$, consider

$$\begin{aligned} v &= v_0, \\ e_1 &= e_{0,1}, \\ e_2 &= e_{0,3}. \end{aligned}$$

By the ascending orientation, $\partial_{v_0}(e_{0,1}) = -1$ and $\partial_{v_0}(e_{0,3}) = -1$.

For $\mathbb{W}^1_{K_{3,3}}$, consider

$$\begin{aligned} e &= e_{0,1}, \\ e_1 &= e_{0,3}, \\ e_2 &= e_{1,2}. \end{aligned}$$

Then $\partial_{v_0}(e_{0,3}) = -1$ and $\partial_{v_1}(e_{1,2}) = -1$ by the ascending orientation.

Given homology cycles

$$\sigma = e_{01} + e_{14} + e_{45} - e_{05}$$

and

$$\tau = e_{03} + e_{34} + e_{45} - e_{05}$$

in $H_1(K_{3,3}, \mathbb{C})$, we can calculate $\mathbb{W}^0_{K_{3,3}}(\sigma \wedge \tau)_{v_0}$ and $\mathbb{W}^1_{K_{3,3}}(\sigma \wedge \tau)_{e_{0,1}}$ as follows:

$$\mathbb{W}^0_{K_{3,3}}(\sigma \wedge \tau)_{v_0} = (-1)(-1)\det \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = 1$$

$$\mathbb{W}^1_{K_{3,3}}(\sigma \wedge \tau)_{e_{0,1}} = (-1)(-1)\det \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = 0.$$

For a homology cycle $\sigma \in H_1(G, \mathbb{C})$, let $\mathrm{Supp}_V(\sigma)$ be the set of vertices in G adjacent to the edges with nonzero coefficients in σ and $\mathrm{Ad}_E(\sigma)$ be the set of edges with nonzero coefficients in σ and the edges adjacent to those in G.

Observation 2.2.13. [CF92] If a vertex v is not covered by the edges with nonzero coefficients in σ and τ, then $\sigma_e = 0$ or $\tau_e = 0$ for any edge e adjacent to v in G. Therefore, we have the following:

1. $\mathbb{W}^0_G(\sigma \wedge \tau)_v = 0$ if $v \notin \mathrm{Supp}_V(\sigma) \cap \mathrm{Supp}_V(\tau)$
2. $\mathbb{W}^1_G(\sigma \wedge \tau)_e = 0$ if $e \notin \mathrm{Ad}_E(\sigma) \cap \mathrm{Ad}_E(\tau)$

We combine the definitions for $\mathbb{W}^0_G$ and $\mathbb{W}^1_G$ to define the Wahl map on a cubic graph:

Definition 2.2.14. [CF92] For a cubic graph G with edge orientation ω and vertex orientation φ, the Wahl map on G is defined as

$$\mathbb{W}_G : \bigwedge^2 H_1(G, \mathbb{C}) \to \mathrm{Cat}_0(G, \mathbb{C}) \oplus \mathrm{Cat}_1(G, \mathbb{C})$$

where $\mathbb{W}_G = \mathbb{W}^0_G \oplus \mathbb{W}^1_G$.

While the definition of $\mathbb{W}_G$ depends on orientations ω, φ, the rank of $\mathbb{W}_G$ is independent of the choice of edge and vertex orientations. We will prove this in Section 2.3.

2.3 Rank Independence

In this section, we will discuss how changing our choice of edge or vertex orientations affects the image of $\mathbb{W}_G$ and how this ultimately affects the rank of $\mathbb{W}_G$.

Note 2.3.1. For a homology cycle $\sigma \in H_1(G, \mathbb{C})$,

$$\partial_v(\sigma) = \sum_{e \ni v} \sigma_e \partial_v(e) = 0$$

for all $v \in \sigma$ because $\partial(\sigma) = 0$.

Recall from Section 2.1 that a graph G must be cubic to define $\mathbb{W}_G$. Then by Note 2.3.1, we know

$$\partial_v(\sigma) = \sigma_{e_1}\partial_v(e_1) + \sigma_{e_2}\partial_v(e_2) + \sigma_{e_2}\partial_v(e_2) = 0 \tag{2.1}$$

for any $v \in V(G)$ with edges e_1, e_2, e_3 adjacent to v. We will use Equation 2.1 numerous times in the subsequent proofs.

The argument for rank independence is a translation of an Italian paper by Ciliberto and Franchetta [CF92]. First, we want to show that for $v \in V(G)$, the value $\mathbb{W}^0_G(\sigma \wedge \tau)_v$ is independent of our choice of e_1 in Definition 2.2.10. Then we will show that $\mathrm{rk}(\mathbb{W}^0_G)$ is independent of vertex orientation φ. Next, we will see how changing vertex orientation φ affects $\mathrm{im}(\mathbb{W}^1_G)$. Finally, we will show the effects on $\mathbb{W}^0_G$ and $\mathbb{W}^1_G$ when we change edge orientation ω.

Lemma 2.3.2. *[CF92] Let $v \in V(G)$ and let φ be a vertex orientation on G. Then $\mathbb{W}^0_G(\sigma \wedge \tau)_v$ is independent of our choice of initial edge in the vertex orientation in Definition 2.2.10.*

Proof. Consider the following:

$$\begin{aligned}
&\partial_v(e_3)\det\begin{pmatrix}\sigma_{e_2} & \tau_{e_2}\\ \sigma_{e_3} & \tau_{e_3}\end{pmatrix} + \partial_v(e_1)\det\begin{pmatrix}\sigma_{e_2} & \tau_{e_2}\\ \sigma_{e_1} & \tau_{e_1}\end{pmatrix}\\
&= \partial_v(e_3)\sigma_{e_2}\tau_{e_3} - \partial_v(e_3)\sigma_{e_3}\tau_{e_2} + \partial_v(e_1)\sigma_{e_2}\tau_{e_1} - \partial_v(e_1)\sigma_{e_1}\tau_{e_2}\\
&= -\tau_{e_2}\Big(\partial_v(e_3)\sigma_{e_3} + \partial_v(e_1)\sigma_{e_1}\Big) + \sigma_{e_2}\Big(\partial_v(e_3)\tau_{e_3} + \partial_v(e_1)\tau_{e_1}\Big).
\end{aligned}$$

By (2.1), this is equal to

$$\begin{aligned}
&= -\tau_{e_2}(-\partial_v(e_2)\sigma_{e_2}) + \sigma_{e_2}(-\partial_v(e_2)\tau_{e_2})\\
&= \partial_v(e_2)\sigma_{e_2}\tau_{e_2} - \partial_v(e_2)\sigma_{e_2}\tau_{e_2} = 0.
\end{aligned}$$

Hence

$$\partial_v(e_3)\det\begin{pmatrix}\sigma_{e_2} & \tau_{e_2}\\ \sigma_{e_3} & \tau_{e_3}\end{pmatrix} = -\partial_v(e_1)\det\begin{pmatrix}\sigma_{e_2} & \tau_{e_2}\\ \sigma_{e_1} & \tau_{e_1}\end{pmatrix} = \partial_v(e_1)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} \tag{2.2}$$

by row swapping in the final determinant. Thus

$$\partial_v(e_1)\partial_v(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} = \partial_v(e_2)\partial_v(e_3)\det\begin{pmatrix}\sigma_{e_2} & \tau_{e_2}\\ \sigma_{e_3} & \tau_{e_3}\end{pmatrix} = \partial_v(e_3)\partial_v(e_1)\det\begin{pmatrix}\sigma_{e_3} & \tau_{e_3}\\ \sigma_{e_1} & \tau_{e_1}\end{pmatrix}.$$

■

Now we will show that $\mathrm{rk}(\mathbb{W}_G^0)$ is independent of the vertex orientation φ. We will do this by showing that changing φ changes all the signs in $\mathrm{im}(\mathbb{W}_G^0)$ consistently.

Lemma 2.3.3. *[CF92] Let v be a vertex in G, let e_1, e_2, e_3 be the edges adjacent to v in some fixed order, and let φ, φ' be two vertex orientations on G. Let $\varphi_v(e_1) = e$ and $\varphi'_v(e_1) = e'$. Then*

$$\partial_v(e)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_e & \tau_e\end{pmatrix} = \partial_v(e')\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e'} & \tau_{e'}\end{pmatrix}$$

if and only if $\varphi = \varphi'$ on v. Otherwise,

$$\partial_v(e)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_e & \tau_e\end{pmatrix} = -\partial_v(e')\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e'} & \tau_{e'}\end{pmatrix}.$$

Proof. Since G is cubic, either $\varphi_v(e_1) = e_2$ or $\varphi_v(e_1) = e_3$ for any vertex orientation φ. Therefore, by (2.2), we have the desired equalities. ■

Unlike with $\mathbb{W}_G^0$, we are not able to say that the rank of $\mathbb{W}_G^1$ is independent of vertex orientation φ. In the following lemma, we show how changing vertex orientations affects $\mathrm{im}(\mathbb{W}_G^1)$.

Lemma 2.3.4. *[CF92] Let $e = (v_1, v_2)$ be an edge in G and let φ, φ' be two vertex orientations on G. Let $\varphi_{v_1}(e) = e_1$, $\varphi'_{v_1}(e) = e'_1$, $\varphi_{v_2}(e) = e_2$, and $\varphi'_{v_1}(e) = e'_2$.*

1. *If $\varphi_{v_1} = \varphi'_{v_1}$ and $\varphi_{v_2} = \varphi'_{v_2}$:*

$$\partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} = \partial_{v_1}(e'_1)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}.$$

2. *If $\varphi_{v_1} \neq \varphi'_{v_1}$ and $\varphi_{v_2} = \varphi'_{v_2}$:*

$$\begin{aligned}\partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} = &-\partial_{v_1}(e'_1)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}\\ &+\partial_{v_2}(e)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_e & \tau_e\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}.\end{aligned}$$

3. *If $\varphi_{v_1} = \varphi'_{v_1}$ and $\varphi_{v_2} \neq \varphi'_{v_2}$:*

$$\begin{aligned}\partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} = &-\partial_{v_1}(e'_1)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}\\ &+\partial_{v_1}(e)\partial_{v_1}(e'_1)\det\begin{pmatrix}\sigma_e & \tau_e\\ \sigma_{e'_1} & \tau_{e'_1}\end{pmatrix}.\end{aligned}$$

4. *If $\varphi_{v_1} \neq \varphi'_{v_1}$ and $\varphi_{v_2} \neq \varphi'_{v_2}$:*

$$\begin{aligned}\partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} &= \partial_{v_1}(e'_1)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}\\ &\quad + \partial_{v_2}(e)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e} & \tau_{e}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}\\ &\quad + \partial_{v_1}(e)\partial_{v_1}(e'_1)\det\begin{pmatrix}\sigma_{e} & \tau_{e}\\ \sigma_{e'_1} & \tau_{e'_1}\end{pmatrix}.\end{aligned}$$

Proof. We will address one case at a time.

1. Since $\varphi_{v_i} = \varphi'_{v_i}$ for $i = 1, 2$, we have $e_i = e'_i$ for $i = 1, 2$. Then by substitution, we get the desired equality.

2. Since $\varphi_{v_2} = \varphi'_{v_2}$, we have $e_2 = e'_2$. By this substitution,

$$\begin{aligned}&- \partial_{v_1}(e'_1)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix} + \partial_{v_2}(e)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e} & \tau_{e}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}\\ &= -\partial_{v_1}(e'_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} + \partial_{v_2}(e)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e} & \tau_{e}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix}\\ &= \partial_{v_2}(e_2)\Big[- \sigma_{e'_1}\tau_{e_2}\partial_{v_1}(e'_1) + \sigma_{e_2}\tau_{e'_1}\partial_{v_1}(e'_1) + \sigma_e\tau_{e_2}\partial_{v_2}(e) - \sigma_{e_2}\tau_e\partial_{v_2}(e)\Big]\\ &= \partial_{v_2}(e_2)\Big[- \tau_{e_2}\Big(\sigma_{e'_1}\partial_{v_1}(e'_1) - \sigma_e\partial_{v_2}(e)\Big) + \sigma_{e_2}\Big(\tau_{e'_1}\partial_{v_1}(e'_1) - \tau_e\partial_{v_2}(e)\Big)\Big].\end{aligned}$$

   Since v_1, v_2 are the endpoints of e, $\partial_{v_1}(e)\sigma_e = -\partial_{v_2}(e)\sigma_e$ and $\partial_{v_1}(e)\tau_e = -\partial_{v_2}(e)\tau_e$. With this substitution, we have

$$\begin{aligned}&\partial_{v_2}(e_2)\Big[- \tau_{e_2}\Big(\sigma_{e'_1}\partial_{v_1}(e'_1) - \sigma_e\partial_{v_2}(e)\Big) + \sigma_{e_2}\Big(\tau_{e'_1}\partial_{v_1}(e'_1) - \tau_e\partial_{v_2}(e)\Big)\Big]\\ &= \partial_{v_2}(e_2)\Big[- \tau_{e_2}\Big(\sigma_{e'_1}\partial_{v_1}(e'_1) + \sigma_e\partial_{v_1}(e)\Big) + \sigma_{e_2}\Big(\tau_{e'_1}\partial_{v_1}(e'_1) + \tau_e\partial_{v_1}(e)\Big)\Big]\\ &= \partial_{v_2}(e_2)\Big[- \tau_{e_2}\Big(- \sigma_{\varphi_{v_1}(e'_1)}\partial_{v_1}(\varphi_{v_1}(e'_1))\Big) + \sigma_{e_2}\Big(- \tau_{\varphi_{v_1}(e'_1)}\partial_{v_1}(\varphi_{v_1}(e'_1))\Big)\Big]\end{aligned}$$

   by (2.1).

   Since $\varphi_{v_1} \neq \varphi'_{v_1}$, we have $e_1 \neq e'_1$, but $\varphi_{v_1}, \varphi'_{v_1}$ are cyclic permutations on three elements that both start at edge e. This means that $\varphi_{v_1}(e_1) = e'_1$ and $\varphi'_{v_1}(e'_1) = e_1$. Then the above equals

$$\begin{aligned}&\partial_{v_2}(e_2)\Big[- \tau_{e_2}\Big(- \sigma_{e_1}\partial_{v_1}(e_1)\Big) + \sigma_{e_2}\Big(- \tau_{e_1}\partial_{v_1}(e_1)\Big)\Big]\\ &= \partial_{v_2}(e_2)\Big[\tau_{e_2}\sigma_{e_1}\partial_{v_1}(e_1) - \sigma_{e_2}\tau_{e_1}\partial_{v_1}(e_1)\Big]\\ &= \partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix}.\end{aligned}$$

3. By a similar argument to Case 2, swapping indices, we get the desired equality.

4. Consider the following:

$$\begin{aligned}
&\partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix}\\
&= \partial_{v_1}(e_1)\partial_{v_2}(e_2)\sigma_{e_1}\tau_{e_2} - \partial_{v_1}(e_1)\partial_{v_2}(e_2)\sigma_{e_2}\tau_{e_1}\\
&= \Big(\partial_{v_1}(e_1)\sigma_{e_1}\Big)\Big(\partial_{v_2}(e_2)\tau_{e_2}\Big) - \Big(\partial_{v_1}(e_1)\tau_{e_1}\Big)\Big(\partial_{v_2}(e_2)\sigma_{e_2}\Big).
\end{aligned}$$

By (2.1), we can substitute to get

$$\begin{aligned}
&\Big(-\partial_{v_1}(e)\sigma_e - \partial_{v_1}(\varphi_{v_1}(e_1))\sigma_{\varphi_{v_1}(e_1)}\Big)\Big(-\partial_{v_2}(e)\tau_e - \partial_{v_2}(\varphi_{v_2}(e_2))\tau_{\varphi_{v_2}(e_2)}\Big)\\
&= -\Big[-\partial_{v_1}(e)\tau_e - \partial_{v_1}\Big(\varphi_{v_1}(e_1)\Big)\tau_{\varphi_{v_1}(e_1)}\Big]\Big[-\partial_{v_2}(e)\sigma_e - \partial_{v_2}\Big(\varphi_{v_2}(e_2)\Big)\sigma_{\varphi_{v_2}(e_2)}\Big].
\end{aligned}$$

Since $\varphi_{v_i} \neq \varphi'_{v_i}$ for $i = 1, 2$, we have $e_i \neq e'_i$, but $\varphi_{v_i}, \varphi'_{v_i}$ are cyclic permutations on three elements that all start at edge e. This means that $\varphi_{v_i}(e_i) = e'_i$ and $\varphi'_{v_i}(e'_i) = e_i$ for $i = 1, 2$. By these substitutions,

$$\begin{aligned}
&-\Big[-\partial_{v_1}(e)\tau_e - \partial_{v_1}\Big(\varphi_{v_1}(e_1)\Big)\tau_{\varphi_{v_1}(e_1)}\Big]\Big[-\partial_{v_2}(e)\sigma_e - \partial_{v_2}\Big(\varphi_{v_2}(e_2)\Big)\sigma_{\varphi_{v_2}(e_2)}\Big]\\
&= \Big[-\partial_{v_1}(e)\sigma_e - \partial_{v_1}(e'_1)\sigma_{e'_1}\Big]\Big[-\partial_{v_2}(e)\tau_e - \partial_{v_2}(e'_2)\tau_{e'_2}\Big]\\
&- \Big[-\partial_{v_1}(e)\tau_e - \partial_{v_1}(e'_1)\tau_{e'_1}\Big]\Big[-\partial_{v_2}(e)\sigma_e - \partial_{v_2}(e'_2)\sigma_{e'_2}\Big]\\
&= \Big[\partial_{v_1}(e)\sigma_e + \partial_{v_1}(e'_1)\sigma_{e'_1}\Big]\Big[\partial_{v_2}(e)\tau_e + \partial_{v_2}(e'_2)\tau_{e'_2}\Big]\\
&- \Big[\partial_{v_1}(e)\tau_e + \partial_{v_1}(e'_1)\tau_{e'_1}\Big]\Big[\partial_{v_2}(e)\sigma_e + \partial_{v_2}(e'_2)\sigma_{e'_2}\Big]\\
&= \partial_{v_1}(e)\sigma_e\partial_{v_2}(e)\tau_e + \partial_{v_1}(e)\sigma_e\partial_{v_2}(e'_2)\tau_{e'_2} + \partial_{v_1}(e'_1)\sigma_{e'_1}\partial_{v_2}(e)\tau_e + \partial_{v_1}(e'_1)\sigma_{e'_1}\partial_{v_2}(e'_2)\tau_{e'_2}\\
&- \partial_{v_1}(e)\tau_e\partial_{v_2}(e)\sigma_e - \partial_{v_1}(e)\tau_e\partial_{v_2}(e'_2)\sigma_{e'_2} - \partial_{v_1}(e'_1)\tau_{e'_1}\partial_{v_2}(e)\sigma_e - \partial_{v_1}(e'_1)\tau_{e'_1}\partial_{v_2}(e'_2)\sigma_{e'_2}.
\end{aligned}$$

Since v_1, v_2 are the endpoints of e, $\partial_{v_1}(e)\sigma_e = -\partial_{v_2}(e)\sigma_e$ and $\partial_{v_1}(e)\tau_e = -\partial_{v_2}(e)\tau_e$. By substitution in the first and fifth terms, the above statement equals

$$\begin{aligned}
&-\partial_{v_1}(e)\sigma_e\partial_{v_1}(e)\tau_e + \partial_{v_1}(e)\sigma_e\partial_{v_2}(e'_2)\tau_{e'_2} + \partial_{v_1}(e'_1)\sigma_{e'_1}\partial_{v_2}(e)\tau_e + \partial_{v_1}(e'_1)\sigma_{e'_1}\partial_{v_2}(e'_2)\tau_{e'_2}\\
&+ \partial_{v_1}(e)\tau_e\partial_{v_1}(e)\sigma_e - \partial_{v_1}(e)\tau_e\partial_{v_2}(e'_2)\sigma_{e'_2} - \partial_{v_1}(e'_1)\tau_{e'_1}\partial_{v_2}(e)\sigma_e - \partial_{v_1}(e'_1)\tau_{e'_1}\partial_{v_2}(e'_2)\sigma_{e'_2}\\
&= \partial_{v_1}(e)\sigma_e\partial_{v_2}(e'_2)\tau_{e'_2} + \partial_{v_1}(e'_1)\sigma_{e'_1}\partial_{v_2}(e)\tau_e + \partial_{v_1}(e'_1)\sigma_{e'_1}\partial_{v_2}(e'_2)\tau_{e'_2}\\
&- \partial_{v_1}(e)\tau_e\partial_{v_2}(e'_2)\sigma_{e'_2} - \partial_{v_1}(e'_1)\tau_{e'_1}\partial_{v_2}(e)\sigma_e - \partial_{v_1}(e'_1)\tau_{e'_1}\partial_{v_2}(e'_2)\sigma_{e'_2}\\
&= \partial_{v_2}(e)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_e & \tau_e\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix} + \partial_{v_1}(e)\partial_{v_1}(e'_1)\det\begin{pmatrix}\sigma_e & \tau_e\\ \sigma_{e'_1} & \tau_{e'_1}\end{pmatrix}\\
&+ \partial_{v_1}(e'_1)\partial_{v_2}(e'_2)\det\begin{pmatrix}\sigma_{e'_1} & \tau_{e'_1}\\ \sigma_{e'_2} & \tau_{e'_2}\end{pmatrix}.
\end{aligned}$$

■

Corollary 2.3.5. *[CF92] The rank of $\mathbb{W}_G$ is independent of vertex orientation φ.*

Proof. Let φ, φ' be two vertex orientations on G. Consider the image of $\mathbb{W}_G$ as coefficient vectors. Then we can form a matrix where a row is the coefficient vector for the image of a particular wedge product of homology cycles and a column is the vector of coefficients for a particular vertex or edge in the image.

The matrix of vectors in the image of $\mathbb{W}_G$ defined by φ' is therefore obtained from the matrix of vectors in the image of $\mathbb{W}_G$ defined by φ by standard column operations, in the following way.

Let $\mathbf{c'}_v$ be the column corresponding to vertex v under orientation φ' and $\mathbf{c}_v$ be the column corresponding to v under orientation φ. By Lemma 2.3.3,

$$\mathbf{c'}_v = \begin{cases} \mathbf{c}_v & \text{if } \varphi_v = \varphi'_v \\ -\mathbf{c}_v & \text{otherwise .} \end{cases}$$

Let $\mathbf{c}_e$ be the column corresponding to edge $e = (u, v)$ under orientation φ' and $\mathbf{c'}_e$ be the column corresponding to e under orientation φ. By Lemma 2.3.4,

$$\mathbf{c'}_e = \begin{cases} \mathbf{c}_e & \text{if } \varphi_u = \varphi'_u, \varphi_v = \varphi'_v \\ -\mathbf{c}_e + \mathbf{c}_v & \text{if } \varphi_u \neq \varphi'_u, \varphi_v = \varphi'_v \\ -\mathbf{c}_e + \mathbf{c}_u & \text{otherwise .} \end{cases}$$

■

Now let's consider how changing the edge orientation ω affects the rank of $\mathbb{W}_G$.

Lemma 2.3.6. *[CF92] The image of $\mathbb{W}_G$ is independent of edge orientation ω.*

Proof. Let ω, ω' be two edge orientation on G. Denote the boundary map defined by ω with ∂ and the boundary map defined by ω' with ∂'. Let $\sigma, \tau \in H_1(G, \mathbb{C})$ be homology cycles under the orientation ω. Then let σ', τ' be homology cycles such that, for each edge e in G,

$$\sigma'_e = \begin{cases} \sigma_e & \text{if } \omega = \omega' \text{ on } e \\ -\sigma_e & \text{if } \omega \neq \omega' \text{ on } e \end{cases}$$

and

$$\tau'_e = \begin{cases} \tau_e & \text{if } \omega = \omega' \text{ on } e \\ -\tau_e & \text{if } \omega \neq \omega' \text{ on } e \end{cases}.$$

For an edge e in G,

$$\partial'_v(e) = \begin{cases} \partial_v(e) & \text{if } \omega = \omega' \text{ on } e \\ -\partial_v(e) & \text{if } \omega \neq \omega' \text{ on } e \end{cases}$$

because there are only two orientations an edge can have. Therefore, we can conclude that $\partial_v(e)\sigma_e = \partial'_v(e)\sigma'_e$ and $\partial_v(e)\tau_e = \partial'_v(e)\tau'_e$. By substitution,

$$\begin{aligned}\partial_v(e_1)\partial_v(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} &= \partial_v(e_1)\partial_v(e_2)\sigma_{e_1}\tau_{e_2} - \partial_v(e_1)\partial_v(e_2)\sigma_{e_2}\tau_{e_1}\\ &= \partial'_v(e_1)\partial'_v(e_2)\sigma'_{e_1}\tau'_{e_2} - \partial'_v(e_1)\partial'_v(e_2)\sigma'_{e_2}\tau'_{e_1}\\ &= \partial'_v(e_1)\partial'_v(e_2)\det\begin{pmatrix}\sigma'_{e_1} & \tau'_{e_1}\\ \sigma'_{e_2} & \tau'_{e_2}\end{pmatrix}\end{aligned}$$

for the map $\mathbb{W}_G^0$ and

$$\begin{aligned}\partial_{v_1}(e_1)\partial_{v_2}(e_2)\det\begin{pmatrix}\sigma_{e_1} & \tau_{e_1}\\ \sigma_{e_2} & \tau_{e_2}\end{pmatrix} &= \partial_{v_1}(e_1)\partial_{v_2}(e_2)\sigma_{e_1}\tau_{e_2} - \partial_{v_1}(e_1)\partial_{v_2}(e_2)\sigma_{e_2}\tau_{e_1}\\ &= \partial'_{v_1}(e_1)\partial'_{v_2}(e_2)\sigma'_{e_1}\tau'_{e_2} - \partial'_{v_1}(e_1)\partial'_{v_2}(e_2)\sigma'_{e_2}\tau'_{e_1}\\ &= \partial'_{v_1}(e_1)\partial'_{v_2}(e_2)\det\begin{pmatrix}\sigma'_{e_1} & \tau'_{e_1}\\ \sigma'_{e_2} & \tau'_{e_2}\end{pmatrix}\end{aligned}$$

for the map $\mathbb{W}_G^1$. ■

Combining Corollary 2.3.5 and Lemma 2.3.6, we get the following independence:

Corollary 2.3.7. *[CF92] The rank of $\mathbb{W}_G$ is independent of the choice of orientations ω, φ.*

As a result of Proposition 2.3.7, we will fix the orientations ω, φ to be the ascending and (123)-orientations for the remainder of our discussion.

Chapter 3 Surjectivity of the Vertex Map $\mathbb{W}_G^0$

3.1 Properties of Cubic Graphs

Since the Wahl map is only defined for cubic graphs, we can use some properties and results for cubic graphs to study $\mathbb{W}_G$. We begin with some results about connectivity.

Menger's Theorem. [Men27] Let G be a graph. Then G is k-edge-connected if and only if there are k edge-disjoint paths between any two vertices of G.

For the purposes of cubic graphs, we are particularly interested in when $k = 3$.

Corollary 3.1.1. *For a graph cycle c in G, let $\mathrm{Supp}_E(c)$ be the set of edges in c. Let G be a cubic, 3-edge-connected graph. For any edge e in G, there exist two graph cycles c_1, c_2 in G such that $e = \mathrm{Supp}_E(c_1) \cap \mathrm{Supp}_E(c_2)$.*

Proof. Let $e = \{a, b\}$ be an edge of G. Since G is cubic, a, b each have exactly three neighboring vertices b, a_1, a_2 and a, b_1, b_2, respectively. By Menger's Theorem, there are three edge-disjoint paths p_1, p_2, p_3 between a, b. Since there are exactly three edges out of a and out of b and three paths between them, e is one of those paths. Without loss of generality, $p_3 = e$. Then, up to relabeling of the neighbors, p_1 goes through a_1, b_1 and p_2 goes through a_2, b_2, as shown in Figure 3.1. Since p_1, p_2, p_3 are edge-disjoint, $c_1 = p_1 \cup p_3$ and $c_2 = p_2 \cup p_3$ are graph cycles where $\mathrm{Supp}_E(c_1) \cap \mathrm{Supp}_E(c_2) = e$. ∎

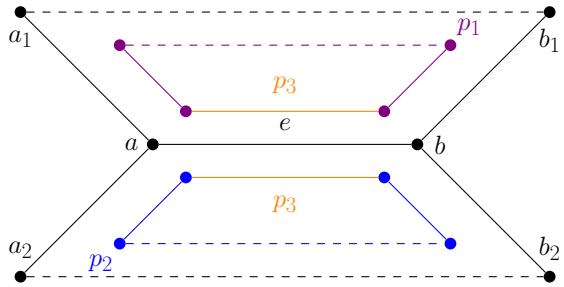


Figure 3.1: Three disjoint paths forming a pair of graph cycles c_1, c_2 that overlap precisely on edge e.

Observation 3.1.2. Let $e = (a, b)$ be an edge in G. Since G is 3-edge-connected there exists a pair of graph cycles c_1, c_2, by Corollary 3.1.1, where e is the only edge in $\mathrm{Supp}_E(c_1) \cap \mathrm{Supp}_E(c_2)$. By our discussion in Section 2.2, we can form homology cycles $\sigma_{c_1}, \sigma_{c_2}$ from c_1, c_2. Therefore, by Observation 2.2.13, $\mathbb{W}_G^0(\sigma_{c_1} \wedge \sigma_{c_2})_v \neq 0$ if and only if $v = a, b$.

In addition to results about connectivity, we need results about planarity.

Kuratowski's Theorem. [Kur30] A graph is planar if and only if it does not contain a subgraph that is a refinement of K_5 or $K_{3,3}$.

Since K_5 has valence 4 vertices and refinements preserve valence, cubic graphs cannot contain a refinement of K_5. However, $K_{3,3}$ is a cubic graph, so any refinement of $K_{3,3}$ has vertices of valence at most 3. Hence planarity of cubic graphs depends only on refinements of $K_{3,3}$. We use this and the subsequent results in our surjectivity proof for $\mathbb{W}_G^0$ in Section 3.2.

Let G be a graph with subgraph H'. Then let

$$\mathrm{proj}_{\mathrm{Cat}_0(H',\mathbb{C})} : \mathrm{Cat}_0(G,\mathbb{C}) \to \mathrm{Cat}_0(H',\mathbb{C})$$

be the projection map and

$$\mathrm{inc}_{H_1(G,\mathbb{C})} : \bigwedge^2 H_1(H',\mathbb{C}) \to \bigwedge^2 H_1(G,\mathbb{C})$$

be the inclusion map. We define the map

$$\mathbb{W}_{G,H'}^0 : \bigwedge^2 H_1(G,\mathbb{C}) \to \mathrm{Cat}_0(H',\mathbb{C})$$

by $\mathbb{W}_{G,H'}^0 = \mathrm{proj}_{\mathrm{Cat}_0(H',\mathbb{C})} \circ \mathbb{W}_G^0$. Then we have $\mathbb{W}_{H'}^0 = \mathbb{W}_{G,H'}^0 \circ \mathrm{inc}_{H_1(G,\mathbb{C})}$ by definition. See Figure 3.2 for a diagram of these maps.

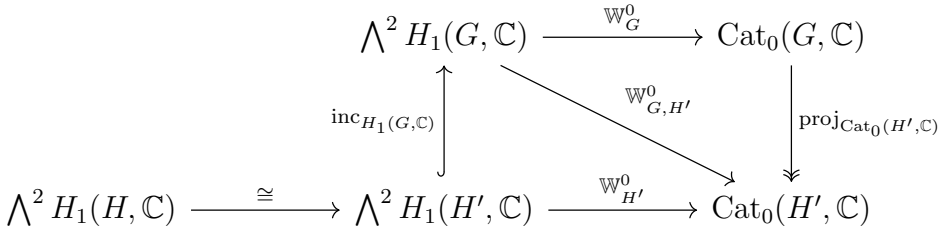


Figure 3.2: The maps defined for Theorem 3.1.3.

Theorem 3.1.3. *Let G, H be cubic graphs. Suppose G is 3-edge-connected and contains a subgraph H' that is isomorphic to a refinement of H. If $\mathbb{W}_H^0$ is surjective, then $\mathbb{W}_{G,H'}^0$ is also surjective.*

Proof. Let H, H' be as in the statement of the Theorem. If $H = H'$, then $\mathbb{W}_{G,H'}^0$ is surjective because $\mathbb{W}_H^0 = \mathbb{W}_{H'}^0 = \mathbb{W}_{G,H'}^0 \circ \mathrm{inc}_{H_1(G,\mathbb{C})}$. Otherwise, let H'' be the graph obtained from H' by smoothing one vertex of valence 2. Let w be that vertex and suppose it is adjacent to vertices a, b, as shown in Figure 3.3.

By induction, we assume that $\mathbb{W}_{G,H''}^0$ is surjective. Since $w \notin \mathrm{Cat}_0(H'',\mathbb{C})$, it suffices to find a pair of cycles $\sigma, \tau \in H_1(G,\mathbb{C})$ such that $\mathbb{W}_G^0(\sigma \wedge \tau)_w \neq 0$ and $\mathbb{W}_G^0(\sigma \wedge \tau)_v = 0$ for all vertices v not in H'.

Recall that G is 3-edge-connected, so by Observation 3.1.2, there exist $\sigma, \tau \in H_1(G,\mathbb{C})$ that overlap precisely on $e = (a, w)$, and $\mathbb{W}_G^0(\sigma \wedge \tau)_v \neq 0$ if and only if $v = a, w$. ∎

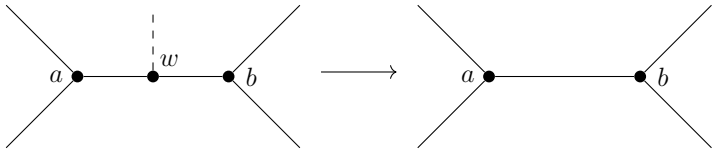


Figure 3.3: Smoothing vertex w to form H''.

Theorem 3.1.4. *Let G, H be cubic graphs. Suppose G is 3-edge-connected and H is a topological minor of G. If $\mathbb{W}^0_H$ is surjective, then $\mathbb{W}^0_G$ is also surjective.*

Proof. Since H is a topological minor of G, G contains a subgraph H' that is isomorphic to a refinement of H. By Theorem 3.1.3, $\mathbb{W}^0_{G,H'}$ is surjective. Now let K be a subgraph of G containing H' such that $\mathbb{W}^0_{G,K}$ is surjective. We prove that $\mathbb{W}^0_G$ is surjective by induction on $|V(K)\backslash V(H')|$ where $V(K)$ is the set of vertices in K and $V(H')$ is the set of vertices in H'.

Let $w \in V(K)\backslash V(H')$ be connected to a vertex a in H' by an edge e. It suffices to find a pair of cycles $\sigma, \tau \in H_1(G, \mathbb{C})$ such that $\mathbb{W}^0_G(\sigma \wedge \tau)_w \neq 0$ and $\mathbb{W}^0_G(\sigma \wedge \tau)_v = 0$ for all vertices v not in $V(H') \cup \{w\}$. By Observation 3.1.2 there exist $\sigma, \tau \in H_1(G, \mathbb{C})$ such that

$$\mathbb{W}^0_G(\sigma \wedge \tau)_v \neq 0$$

if and only if $v = a, w$. ■

3.2 Surjectivity Criteria

Given that a graph G is cubic and 3-edge-connected, Miranda finds the following result about surjectivity of $\mathbb{W}^0_G$:

Theorem 3.2.1. *[Mir89] Let G be a cubic, 3-edge-connected graph. If G is planar, then $\mathbb{W}^0_G$ is not surjective.*

We prove that the converse also holds, using results from Section 3.1.

Theorem 3.2.2. *Let G be a cubic, 3-edge-connected graph. If G is non-planar, then $\mathbb{W}^0_G$ is surjective.*

Proof. By Kuratowski's Theorem, G contains a subgraph that is a refinement of $K_{3,3}$. Therefore, by Theorem 3.1.4 it suffices to show that $\mathbb{W}^0_{K_{3,3}}$ is surjective.

Since the rank of $\mathbb{W}^0_G$ is independent of vertex or edge orientations, we choose the ascending and (123)-orientations. Consider the cycle set

$$\begin{aligned}\mathcal{B} = \{&\sigma_1 = e_{1,4} + e_{4,5} + e_{0,5} - e_{0,1}\\ &\sigma_2 = e_{3,4} + e_{4,5} + e_{0,5} - e_{0,3}\\ &\sigma_3 = e_{1,2} + e_{2,5} + e_{0,5} - e_{0,1}\\ &\sigma_4 = e_{2,3} + e_{2,5} + e_{0,5} - e_{0,3}\}\end{aligned}$$

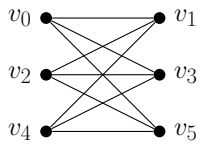


Figure 3.4: This is the vertex labeling of $K_{3,3}$ used in this proof.

on $K_{3,3}$ where the vertices of $K_{3,3}$ are labeled as in Figure 3.4. Then

$$\begin{array}{rcrcr}
\mathbb{W}^0_G(\sigma_1 \wedge \sigma_2) & = & v_0 & + & v_4 \\
\mathbb{W}^0_G(\sigma_1 \wedge \sigma_3) & = & v_1 & - & v_5 \\
\mathbb{W}^0_G(\sigma_1 \wedge \sigma_4) & = & v_0 & - & v_5 \\
\mathbb{W}^0_G(\sigma_2 \wedge \sigma_3) & = & -v_0 & - & v_5 \\
\mathbb{W}^0_G(\sigma_2 \wedge \sigma_4) & = & -v_3 & - & v_5 \\
\mathbb{W}^0_G(\sigma_3 \wedge \sigma_4) & = & v_0 & - & v_2.
\end{array}$$

Note that v_1, v_2, v_3, v_4 appear exactly once and $\mathbb{W}^0_G(\sigma_1 \wedge \sigma_4), \mathbb{W}^0_G(\sigma_2 \wedge \sigma_3)$ have the same vertices with nonzero coefficients but with opposite signs. Hence this is a linearly independent set, so $\mathbb{W}^0_{K_{3,3}}$ is surjective. ■

Chapter 4 Surjectivity of the Overall Map $\mathbb{W}_G$

4.1 Girth of Graphs

We found in Section 2.3 that $\mathrm{im}(\mathbb{W}_G)$ depends on the choice of vertex orientation φ. Therefore, it is difficult to study the image of $\mathbb{W}_G|_{\ker \mathbb{W}^0}$ with a fixed vertex orientation, like the (123)-orientation. Instead, we return to the work of Ciliberto and Franchetta [CF92]. They find a family of graphs for which the Wahl map is surjective. One of the properties of these graphs is that they all have girth at least 5. We show that this condition is necessary for the Wahl map to be surjective.

Theorem 4.1.1. *If $\mathbb{W}_G$ is surjective, then* $\mathrm{girth}(G) \geq 5$.

Proof. Let G be a graph of girth k. Consider a k-cycle σ in G with vertices $v_1, \ldots, v_k$ and edges

$$\begin{aligned} e_1 &= (v_1, v_2), \\ &\cdots \\ e_{k-1} &= (v_{k-1}, v_k), \\ e_k &= (v_1, v_k). \end{aligned}$$

Let $T_k = \bigcup_{i=1}^{k} \{e_i, v_i\}$ and

$$\mathrm{proj}_{T_k} : \mathrm{Cat}_0(G) \oplus \mathrm{Cat}_1(G) \to \mathrm{Cat}_0(T_k) \oplus \mathrm{Cat}_1(T_k)$$

be the projection map. It suffices to show that the composition $\mathrm{proj}_{T_k} \circ \mathbb{W}_G$ is not surjective for $k \leq 4$.

Since G is cubic, each of $v_1, \ldots, v_k$ have one additional neighbor not in T_k. We label those neighbors $w_1, \ldots, w_k$ with

$$\begin{aligned} e_{k+1} &= (v_1, w_1), \\ &\cdots \\ e_{2k} &= (v_k, w_k). \end{aligned}$$

Consider the 1-chains of the form

$$\tau_i = e_{k+i} + e_i - e_{k+i+1}$$

for $1 \leq i \leq k-1$ and $\tau_k = e_{2k} + e_k - e_{k+1}$. Then the restrictions of any cycle to T_k can be written as a linear combination of the τ_i. This is because τ_i covers exactly one edge, e_i, in T_k and the edges adjacent to e_i in G that connect T_k to the rest of G. Thus

$$\mathrm{rk}(\mathrm{proj}_{T_k} \circ \mathbb{W}_G) \leq \binom{k}{2}.$$

Since $\dim(\mathrm{Cat}_0(T_k) \oplus \mathrm{Cat}_1(T_k)) = 2k$, the composition $\mathrm{proj}_{T_k} \circ \mathbb{W}_G$ cannot be surjective if $\binom{k}{2} < 2k$. Hence if $k < 5$, the map $\mathbb{W}_G$ is not surjective. ∎

Example 4.1.2. Suppose G contains a 3-cycle σ with vertices v_1, v_2, v_3 and edges $e_1 = (v_1, v_2)$, $e_2 = (v_2, v_3)$, $e_3 = (v_1, v_3)$. Let $T_3 = \{e_1, e_2, e_3, v_1, v_2, v_3\}$. Since G is cubic, each of v_1, v_2, v_3 have one additional neighbor not in T_3. We label those neighbors w_1, w_2, w_3 with $e_4 = (v_1, w_1)$, $e_5 = (v_2, w_2)$, $e_6 = (v_3, w_3)$, as shown in Figure 4.1.

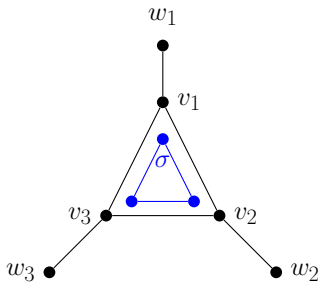


Figure 4.1: The 3-cycle σ and its neighbors as labeled in the proof.

There are seven different ways that a cycle τ_i can go through T as shown in Figure 4.2:

$$\begin{aligned} \sigma &\text{ through } e_1, e_2, e_3 \\ \tau_1 &\text{ through } e_4, e_1, e_2, e_6 \\ \tau_2 &\text{ through } e_5, e_2, e_3, e_4 \\ \tau_3 &\text{ through } e_6, e_3, e_1, e_5 \\ \tau_4 &\text{ through } e_6, e_3, e_4 \\ \tau_5 &\text{ through } e_4, e_1, e_5 \\ \text{and } \tau_6 &\text{ through } e_5, e_2, e_6. \end{aligned}$$

Notice that the cycles $\sigma, \tau_1, \ldots, \tau_6$ are linear combinations of τ_4, τ_5, τ_6, as demonstrated in Figure 4.3. Then there are $\binom{3}{2} = 3$ ways to form cycle wedge pairs from τ_4, τ_5, τ_6. Hence, the rank of $\mathbb{W}_G$ on T_3 is at most $3 < 6 = \dim(\mathrm{Cat}_0(T_3) \oplus \mathrm{Cat}_1(T_3))$.

The converse of Theorem 4.1.1 is not true. For example, the Heawood graph H_{14} (Figure 4.4) is a cubic, 3-edge-connected, non-planar graph of girth 6, but $\mathbb{W}_{H_{14}}$ is not surjective. Note that H_{14} has genus 8, so

$$\dim(\bigwedge^2 H_1(H_{14}, \mathbb{C})) = \binom{8}{2} = 28$$

and

$$\dim(\mathrm{Cat}_0(H_{14}, \mathbb{C}) \oplus \mathrm{Cat}_1(H_{14}, \mathbb{C})) = 5 * 8 - 5 = 35.$$

Therefore, since $35 > 28$, no map $\bigwedge^2 H_1(H_{14}, \mathbb{C}) \to \mathrm{Cat}_0(H_{14}, \mathbb{C}) \oplus \mathrm{Cat}_1(H_{14}, \mathbb{C})$ can be surjective, particularly $\mathbb{W}_{H_{14}}$. Despite this, we will see in the next section that most cubic graphs seem to be surjective if they are girth at least 5 and are sufficiently large.

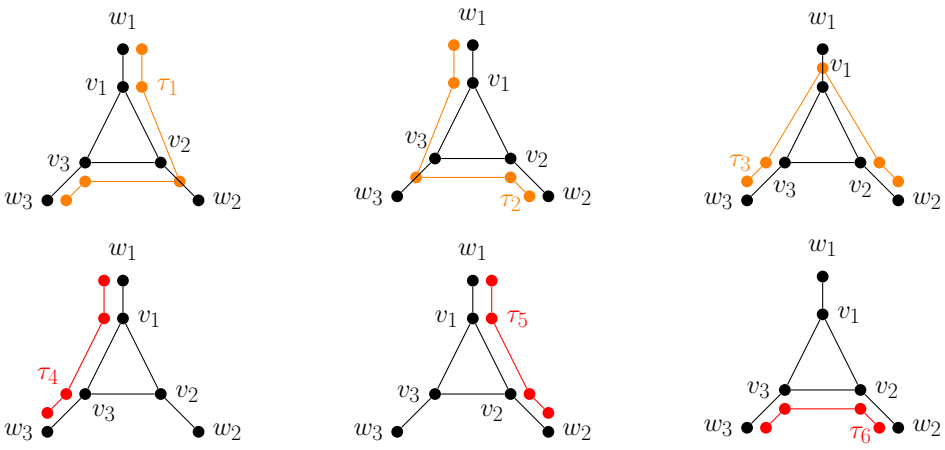


Figure 4.2: The cycles through T_3.

4.2 Programming Results

If we generate a random, cubic graph G with girth at least 5, how likely is it that $\mathbb{W}_G$ is surjective? I wrote a program in SageMath to help answer this question. See Appendix A for more details. We have the following data in Table 4.1 from this program.

Table 4.1: A sampling of random cubic graphs with girth at least 5, of various sizes.

Number of Vertices	Number of Samples	Number for which $\mathbb{W}_G$ is Surjective	Proportion
50	100	90	0.90
100	100	96	0.96
150	100	96	0.96
200	100	97	0.97
250	100	99	0.99
300	100	99	0.99
350	100	100	1.00
400	100	100	1.00

This data that suggests the following conjecture:

Conjecture 4.2.1. *Let F_g be the event that $\mathbb{W}_G$ is surjective for a random cubic graph G of genus g and* girth$(G) \geq 5$. *Then we have*

$$\lim_{g\to\infty} P(F_g) = 1.$$

Since cubic graphs are almost always 3-edge-connected [RW92], there is a high probability that this criterion is met. Similarly, as the number of vertices grows, the chance that a graph is planar goes to zero [BB01].

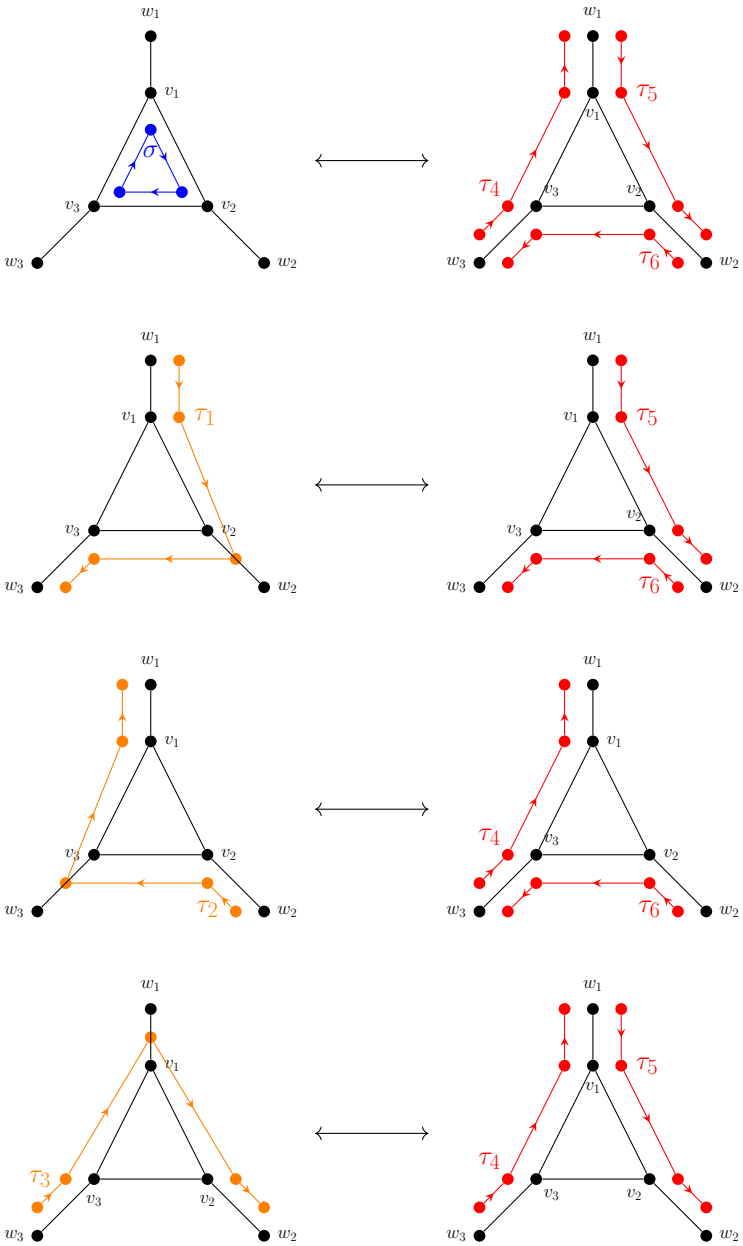


Figure 4.3: The linear combinations of τ_4, τ_5, τ_6 to form $\sigma, \tau_1, \tau_2, \tau_3$.

It is important to note that while the probability in Conjecture 4.2.1 may approach 1, it will never equal 1. This is because $\mathbb{W}_G$ is not surjective for a planar graph G, so we can always produce a counterexample.

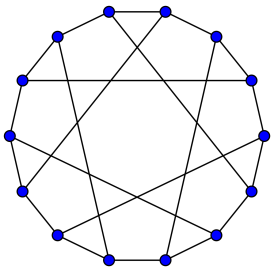


Figure 4.4: The Heawood graph.

Appendices

Appendix A: SageMath Program

I wrote the following program to evaluate the Wahl map on a given cubic graph and to generate random cubic graphs of girth at least 5. The code can be accessed and run at the following link: `https://github.com/hansonac13/Wahl_map.git`

All graphs should be entered as Graph({vertex1:[list_of_v1_neighbors], vertex2:[list_of_v2_neighbors], ...}) or some other acceptable graph object according to SageMath.

To evaluate the Wahl map on a graph G, use build_matrix(G). This outputs a matrix where the rows correspond to cycle wedge pairs and the columns correspond to each vertex in ascending order and then each edge in lexicographic order. To check if $\mathbb{W}_G$ surjective, use is_surj(G). The output to is_surj() is a printed statement. Similarly, you can check if $\mathbb{W}_G^0$ is surjective with is_V_surj(G). It will also print a statement.

In order to generate n non-isomorphic, random cubic graphs with m vertices of girth at least 5, run the command sample_random_girth5(n,m). If you want to make these graphs truly random for probabilistic purposes, add the entry noniso=False at the end of the input list. The output will be a list of graphs.

To test and store examples, run add_to_files(examples) where examples is a list of graphs.

```
from __future__ import absolute_import
from sympy import *
import sys
from sage.all import *
from sage.graphs import graph
from sage.graphs.generic_graph import *


###### This section of the code evaluates the Wahl map on a given
###### graph.
def pair_up_cycles(cycleSet):
    '''This will produce pairs of cycles from the set of cycles
    in the order in which they appear in the set. The pairs are
    returned as a list of pairs where each pair is a list of two
    cycles. Cycles are represented as a list of integers which
    correspond to vertex labels.'''
    pairs=list()

    for i in range(len(cycleSet)):
        for j in range(i+1,(len(cycleSet))):
            pairs.append([cycleSet[i],cycleSet[j]])
```

```
    return(pairs)


def cycle_orientation(v, neighbor_pt, cycle):
    '''This will take a vertex v and one of its neighbors
    (neighbor_pt) and calculates whether the cycle runs into
    vertex v along that neighbor or out of the vertex along that
    neighbor. The output of this function is 1 (into) or -1
    (out of).'''

    if cycle.index(v) == 0 and cycle.index(neighbor_pt) == len(cycle)-1:
        '''This accounts for if the neighbor is the last point in
        the cycle list and v is the first in the cycle list, so
        neighbor_pt comes before v.'''
        return(1)
    elif cycle.index(v) == len(cycle)-1 and cycle.index(neighbor_pt) == 0:
        '''This accounts for if the neighbor is the first point
        in the cycle list and v is the last in the cycle list,
        so neighbor_pt comes after v.'''
        return(-1)
    elif cycle.index(v) == cycle.index(neighbor_pt)-1:
        '''Here v and its neighbor are not at the end or
        beginning of the cycle list, and v comes before
        neighbor_pt.'''
        return(-1)
    elif cycle.index(v)-1 == cycle.index(neighbor_pt):
        '''Here v and its neighbor are not at the end or
        beginning of the cycle list, and v comes after
        neighbor_pt.'''
        return(1)
    else:
        '''This is a catch-all in case neighbor_pt and v are not
        actually both in this cycle consecutively.'''
        return(0)


def edge_orientation(v, neighbor_pt):
    '''This will take a vertex v and one of its neighbors
    (neighbor_pt) and calculates whether the edge orientation runs
    into vertex v along the neighbor or out of the vertex along
    the neighbor. The output of this function is 1 (into) or -1
    (out of). The edge orientation is, by default, directed from
    lowest vertex label to highest. Since vertices are labelled
    by the natural numbers without repetition, this orientation
    is well-defined.'''
```

```
    if v < neighbor_pt:
        '''Since v is smaller than neighbor_pt, it comes before,
        so the edge between them is oriented out of v.'''
        return(-1)
    elif v > neighbor_pt:
        '''Since v is larger than neighbor_pt, it comes after,
        so the edge between them is oriented into v.'''
        return(1)
    else:
        print("Error with finding neighbor edge value. Check input.")


def w_vertex(v, cycle1, cycle2, theGraph):
    '''This takes a vertex v and pair of cycles, cycle1 and
    cycle2, and calculates the Wahl map of the wedge of those
    two cycles on v. This involves calculating the determinant
    of a 2x2 matrix where the first row corresponds to cycle1
    and the second to cycle2 and then the columns correspond
    to two of the neighboring vertices to v. Since these
    graphs are cubic, there are exactly three neighbors to
    each vertex. We will select two neighbors according to
    the permutation phi which is (123), where 1, 2, and 3
    correspond to the first, second, and third neighbors of v
    in ascending order. This does not effect the rank of the
    Wahl map because rank is independent of vertex orientation.
    It returns the determinant value times the edge orientation
    values for each neighboring edge used in the determinant
    calculation. This product can equal -1, 0, or 1.'''
    neighborlist = theGraph.neighbors(v)
    n1 = min(neighborlist[0],neighborlist[1],neighborlist[2])

    '''Here we find the second neighbor based on the vertex
    orientation phi. Since phi has three elements for the
    three neighbors, we want the next neighbor to be next in
    the list modulo 3.'''
    index1 = neighborlist.index(n1)
    index2 = Mod(index1+1,3)
    n2 = neighborlist[index2]

    '''This starts the determinant calculation where aij is the
    ith row and jth column entry. The value comes from the cycle
    orientation from v to its neighbor. So a12 is the
    orientation value from v to n2 in cycle1. If the relevant
    neighbor is not in the relevant cycle, that matrix entry is
    0.'''
    if (v in cycle1) and (v in cycle2):
        if (n1 in cycle1):
```

```
            a11 = cycle_orientation(v,n1,cycle1)
        else:
            a11 = 0
        if (n2 in cycle1):
            a12 = cycle_orientation(v,n2,cycle1)
        else:
            a12 = 0
        if (n1 in cycle2):
            a21 = cycle_orientation(v,n1,cycle2)
        else:
            a21 = 0
        if (n2 in cycle2):
            a22 = cycle_orientation(v,n2,cycle2)
        else:
            a22 = 0

        '''This is where we calculate the edge orientation value
        of 1 or -1 for towards or away from v, respectively.'''
        delta1 = edge_orientation(v, n1)
        delta2 = edge_orientation(v, n2)
        det = delta1*delta2*(a11*a22 - a12*a21)
    else:
        '''If v is in neither cycle1 nor cycle2, then the
        matrix determinant is trivially 0.'''
        det = 0

    return(det)



def w_edge(edge, cycle1, cycle2, theGraph):
    '''This takes an edge=(v1 and v2) and pair of cycles,
    cycle1 and cycle2, and calculates the Wahl map of the
    wedge of those two cycles on that edge. This involves
    calculating the determinant of a 2x2 matrix where the
    first row corresponds to cycle1 and the second to
    cycle2 and then the first column corresponds to a
    neighboring edge to v1 of edge and the second column
    to a neighboring edge of v2 of edge. Since these graphs
    are cubic, there are two neighbors to each vertex. We
    will select the neighbor with minimal vertex label for
    consistency. This does not effect the rank of the Wahl
    map since it is the same as the choice of vertex
    orientation. It returns the determinant value times the
    edge orientation values for each neighboring edge used
    in the determinant calculation. This product can equal
    -2, -1, 0, 1, or 2.'''
```

```
endpoints = list(edge)
v1 = endpoints[0]
v2 = endpoints[1]

'''Here we find the neighbors of each endpoint, v1 and v2,
including the other endpoint of the given edge. Since these
graphs are trivalent, there are three vertex neighbors.'''
neighborlist1 = theGraph.neighbors(v1)
neighborlist2 = theGraph.neighbors(v2)

'''This chooses the neighbors n1 for v1 and n2 for v2
according to the permutation phi. To make sure the
permutation does not return the other endpoint of the edge
we are referencing, we make the other endpoint the starting
vertex in the permutation and choose the neighbor that is
next in the orientation.'''
index1 = neighborlist1.index(v2)
index2 = neighborlist2.index(v1)
'''Here we find the next neighbor of v1, after v2, based on
the vertex orientation phi. Since phi has three elements for
the three neighbors, we want the next neighbor to be next in
the list modulo 3.'''
index3 = Mod(index1+1,3)
n1 = neighborlist1[index3]
'''Here we find the next neighbor of v2, after v1, based on
the vertex orientation phi. Since phi has three elements for
the three neighbors, we want the next neighbor to be next in
the list modulo 3.'''
index4 = Mod(index2+1,3)
n2 = neighborlist2[index4]

'''This starts the determinant calculation where aij is the
ith row and jth column entry. The value comes from the cycle
orientation from v1 to n1 and from v2 to n2. So a12 is the
orientation value from v2 to n2 in cycle1. If the relevant
neighbor is not in the relevant cycle, that matrix entry is
0.'''
if (v1 in cycle1) and (n1 in cycle1):
    a11 = cycle_orientation(v1,n1,cycle1)
else:
    a11 = 0
if (v2 in cycle1) and (n2 in cycle1):
    a12 = cycle_orientation(v2,n2,cycle1)
else:
    a12 = 0
if (v1 in cycle2) and (n1 in cycle2):
    a21 = cycle_orientation(v1,n1,cycle2)
```

```
    else:
        a21 = 0
    if (v2 in cycle2) and (n2 in cycle2):
        a22 = cycle_orientation(v2,n2,cycle2)
    else:
        a22 = 0

    '''This is where we calculate the edge orientation value of
    1 or -1 for towards or away from v1 and v2, respectively.'''
    delta1 = edge_orientation(v1, n1)
    delta2 = edge_orientation(v2, n2)

    return(delta1*delta2*(a11*a22 - a12*a21))


def generate_row(cycle1, cycle2, theGraph):
    '''This function takes in two cycles, cycle1 and cycle2,
    and returns a vector which contains the Wahl map values
    on each vertex in ascending order and then on each edge
    in lexicographic order.'''
    vect=list()

    for v in theGraph.vertices():
        entry = w_vertex(v, cycle1,cycle2, theGraph)
        vect.append(entry)
    for e in theGraph.edges():
        entry = w_edge(e, cycle1, cycle2, theGraph)
        vect.append(entry)

    return(vector(ZZ,vect))


def build_matrix(G):
    '''Here the the graph G is used to calculate the matrix of
    Wahl map value vectors for all cycle wedge pairs.'''

    '''The list of cycles we use is the cycle basis determined
    by SageMath.'''
    pairs=pair_up_cycles(G.cycle_basis())

    '''We need an empty matrix of the proper dimension to fill
    with Wahl map values. This matrix has as many rows as cycle
    pairs and as many columns as vertices and edges.'''
    edgeMatrix=matrix(ZZ,len(pairs),len(G.vertices())+len(G.edges()))

    '''Here we find the vector of Wahl map values for each
    cycle wedge pair and insert them in the matrix as rows.'''
```

```
    for i in range(len(pairs)):
        edgeMatrix.set_row(i,generate_row(pairs[i][0],pairs[i][1],G))

    return(edgeMatrix)


def is_surj(G):
    '''This takes in a graph and prints whether or not the
    Wahl map is surjective on that graph.'''
    M=build_matrix(G)

    if M.rank() == M.ncols():
        print("The Wahl map is surjective on this graph.")
    else:
        print("The Wahl map is NOT surjective on this graph.")
        print(M.rank(), M.ncols())


def is_V_surj(G):
    '''This takes in a graph and prints whether or not the vertex
    map V is surjective on that graph. Note that the Wahl map on
    a graph equals V+E where V is the map restricted to acting on
    the vertices, and E is the map restricted to acting on the
    edges.'''
    M=build_matrix(G)

    '''Here we build a list of indices for the rows and columns
    of the image of V.'''
    r = list()
    for i in range(M.nrows()):
        r.append(i)
    c = list()
    for j in range(len(G.vertices())):
        c.append(j)

    '''This builds a submatrix of the Wahl matrix of values to
    only keep the columns corresponding to vertex values.'''
    vM = M.matrix_from_rows_and_columns(r, c)

    if vM.rank() == vM.ncols():
        print("The vertex map is surjective on this graph.")
    else:
        print("The vertex map is NOT surjective on this graph.")
        print(vM.rank(), vM.ncols())


###### This section of the code will generate random cubic
```

```
###### graphs of minimum girth 5.
def generate(numVertices, girth=5):
    '''This function intakes the number of vertices desired and
    returns a cubic graph of with that many vertices and girth
    at least 5.'''
    n = numVertices
    '''We need an empty graph to add edges and vertices to.'''
    G = Graph()

    '''This double checks that there are at least the girth
    number of vertices.'''
    if n < girth:
        print("You have not given enough vertices.")
    else:
        '''Here we build the vertex set labeled 0 to n-1 and
        form the primary cycle 0 up to n-1 and back to 0. This
        makes all vertices valence 2 so we add exactly more
        edge per vertex in the next step to be cubic.'''
        for v in range(n-1):
            G.add_edge(v,v+1)
        G.add_edge(n-1,0)

        remaining_vertices = [i for i in range(0,n)]
        for v1 in remaining_vertices:
            '''This stops any infinite loops looking for girth
            big enough when it is not possible. We do this by
            working from a list specific to v1.'''
            possible = remaining_vertices.copy()

            '''We remove vertices adjacent to v1 in the original
            cycle so that we remain simple. We also remove
            vertices within girth-1 of v1 to avoid obvious girth
            violations.'''
            for j in range(girth):
                if Mod(v1+j,n) in possible:
                    possible.remove(Mod(v1+j,n))
                if Mod(v1-j,n) in possible:
                    possible.remove(Mod(v1-j,n))

            '''We want to keep looking for a neighbor for v1
            until it is degree 3 or we run out of vertices to
            pair with it.'''
            while G.degree(v1) < 3 and len(possible) > 0:

                '''Now we pick the second vertex to form an edge with.'''
                v2_index = sage.misc.prandom.randint(0,len(possible)-1)
                v2 = possible[v2_index]
```

```
                G.add_edge(v1,v2)
                if G.girth() < girth or G.degree(v2) > 3:
                    '''If girth is too small, we remove the edge and
                    drop v2 from the potential neighbors of v1.'''
                    G.delete_edge(v1,v2)
                    possible.remove(v2)
                else:
                    '''If girth is high enough, we remove v2 from the
                    vertex list so that valence of v2 stays at 3.'''
                    remaining_vertices.remove(v2)

        if G.is_regular() and G.girth()==5:
            return(G)
        else:
            return(Graph())



def sample_random_girth5(n,vcount,check_planarity=False,check_tri=False):
    '''Here we choose a number n of random cubic graphs we want to
    test, and it returns a list of n cubic graphs. If noniso=True,
    then these graphs will be non-isomorphic. Otherwise,
    the graphs will be truly random, with possible repetition.
    We can specify the number of vertices or have a number selected
    at random. We also have the option to check for non-planar,
    3-vertex-connected graphs or not. By default this function does
    not make those checks.'''
    sample = list()
    graphset = list()
    total_number = 0

    while total_number < n:
        G = generate(vcount)
        new = True
        if G != Graph():

            '''This skips graphs that are the same up to isomorphism
            as ones we seen before in a single test run'''
            if noniso:
                for H in sample:
                    if H.is_isomorphic(G):
                        new = False

            if new == True:
                sample.append(G)
```

```
                    if check_planarity and check_tri:
                        if G.is_planar() == False and G.is_tri():
                            '''These are 3-vertex-connected and non-planar
                            graphs which we generally want. They are also
                            of girth at least 5 which seems to be
                            necessary for surjectivity.'''
                            graphset.append(G)
                            total_number += 1
                    elif check_planarity:
                        if G.is_planar() == False:
                            '''These are girth at least 5, non-planar
                            graphs, but they may not be 3-connected.'''
                            graphset.append(G)
                            total_number += 1
                    elif check_tri:
                        if G.is_tri():
                            '''These are girth at least 5, 3-connected
                            graphs, but they may not be non-planar.'''
                            graphset.append(G)
                            total_number += 1
                    else:
                        '''These are graphs of girth at least 5.'''
                        graphset.append(G)
                        total_number += 1

    return(graphset)

def add_to_file(examples):
    '''This function creates a file and stores the graphs in the list
    of examples. It also records which of them the Wahl map is
    surjective on.'''
    lines = list()

    '''Here we build a list of which of the graphs the Wahl map is
    surjective on.'''
    isSet = list()
    for g in examples:
        M=build_matrix(g)
        if M.rank() == M.ncols():
            isSet.append(examples.index(g))

    print(str(len(isSet))," out of ",str(len(examples)))
    '''This puts how many graphs the Wahl map is surjective on, how
    many are in the example list, and the indexes of the graphs the
    Wahl map is surjective on.'''
    lines.append(str(len(isSet)))
    lines.append(str(len(examples)))
```

```
lines.append(str(isSet))

for ex in examples:
    '''This is where the examples are recorded as an index, its
    girth, and its list of edges.'''
    lines.append(str(examples.index(ex)))
    lines.append(str(ex.girth()))
    lines.append(str(ex.edges()))

with open('RandomExamples.sagews', 'w') as f:
    '''Here is where the file is created and written in with the
    information above.'''
    f.write('\n'.join(lines))
```

Bibliography

[BB01] B. Bollobás and B. Béla. *Random Graphs*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2001.

[CF92] C. Ciliberto and A. Franchetta. L'applicazione di Gauss per grafi trivalenti. *Rendiconti di Matematica*, 12(7):165 – 196, 1992.

[CHM88] Ciro Ciliberto, Joe Harris, and Rick Miranda. On the surjectivity of the Wahl map. *Duke Mathematical Journal*, 57(3):829 – 858, 1988.

[Kur30] Kazimierz Kuratowski. Sur le probleme des courbes gauches en topologie. *Fundamenta Mathematicae*, 15:271 – 283, 1930.

[Men27] Karl Menger. Zur allgemeinen Kurventheorie. *Fundamenta Mathematicae*, 10:96 – 115, 1927.

[Mir89] Rick Miranda. The gaussian map for certain planar graphs. *Preprint*, 1989.

[RW92] R. W. Robinson and N. C. Wormald. Almost all cubic graphs are hamiltonian. *Random Structures amp; Algorithms*, 3(2):117–125, 1992.

[Wah87] Jonathan M Wahl. The jacobian algebra of a graded gorenstein singularity. *Duke Mathematical Journal*, 55(4):843–871, 1987.

Curriculum Vita

Angela C. Hanson

Place of Birth:

- Fort Leonard Wood, MI

Education:

- University of Kentucky, Lexington, KY
  M.A. in Mathematics, Dec. 2019
- Rose-Hulman Institute of Technology, Terre Haute, IN
  B.S. in Mathematics, May. 2017

Professional Positions:

- Graduate Teaching Assistant, University of Kentucky, Fall 2018–Spring 2023
- Graduate Teaching Assistant, Eastern Illinois University, Fall 2017–Spring 2018

Honors

- College of Arts & Sciences Outstanding TA Award, University of Kentucky
- Department of Mathematics Diversity, Equity and Inclusion Award, University of Kentucky
- Emeritus Faculty Endowed Doctoral Fellow, University of Kentucky
- Ed Enochs Graduate Scholarship in Algebra, University of Kentucky
- Graduate Scholars in Mathematics Fellow, University of Kentucky

Publications & Preprints:

- "Implementing student-centered assessment: motivations, experiences, supports, and challenges" with Justin Barhite, Ben Braun, Courtney George, Hunter Lehmann, Camille Schuetz, and Chloe Wawrzyniak (Submitted)
- "Sliding Block Puzzles with a Twist: On Segerman's 15+4 Puzzle" with Patrick Garcia, David Jensen, and Noah Owen (Submitted)
- Internal Department of Defense Paper, Ft. Meade, Maryland, Aug. 2015