university of groningen

faculty of science and engineering

mathematics and applied mathematics

Computing Arakelov–Green Functions on Metrized Graphs

Bachelor's Project Mathematics

July 2021

Student: R.M. van Dijk

First supervisor: Dr. J.S. Müller

Second assessor: O. Lorscheid

With special thanks to E. Kaya

Abstract

Arakelov–Green functions on metrized graphs have important applications in number theory, and it is therefore useful to have an easy method of computing their values. This thesis provides an introduction to metrized graphs and Arakelov–Green functions, before recalling several of Zubeyir Cinkir's findings in this field. We use these results to construct formulas for Arakelov–Green functions with respect to any admissible metric, which allow us to devise an efficient algorithm to compute the functions' values. We conclude by applying this algorithm to five different examples using an implementation in Sage.

Contents

1 Introduction

Metrized graphs are interesting for many fields of science, including circuit design, neurobiology, quantum physics and tropical geometry [1]. In particular, certain functions on these metrized graphs, known as Arakelov–Green functions, have important applications in number theory, as explained by Zhang in [17].

In [8], Zubeyir Cinkir gives a set of formulas for the Arakelov–Green function $g_{\mu_{can}}$ with respect to the canonical measure, which in turn allowed him to design an efficient algorithm to compute its values. He also suggests an approach to extend these formulas so that it can be used to compute g_{μ_D}, which is a generalization of $g_{\mu_{can}}$ introduced in by Zhang in [17]. However, Cinkir does not give any concrete expressions or proofs for this method.

The main goal of this Bachelor's thesis is to determine these formulas for g_{μ_D}. Then, with these expressions, we devise an algorithm to efficiently compute values of g_{μ_D}, and implement it in Sage.

The remainder of this section recalls preliminaries from graph and measure theory. Section 2 details the fundamental notions related to metrized graphs, along with some elementary but important results. Section 3 discusses ideas from calculus, such as differentiation and smoothness, but applies them to functions on metrized graphs. Section 4 considers two such important functions, called the voltage and resistance functions. Theorem 4.8 summarizes one of Cinkir's findings concerning the resistance function.

Section 5 defines Arakelov–Green functions, gives Cinkir's main result in Theorem 5.9, and extends it in Theorem 5.20. Section 6 uses Theorem 5.20 for an algorithm to compute values of g_{μ_D}, which we apply to a number of examples in Section 7. Specifically, Section 7.4 discusses the epsilon invariant and provides a method to compute it.

1.1 Graph theory in brief

A graph is, loosely speaking, a collection of points (vertices) along with a collection of links between pairs of vertices (edges). More concretely, a **simple** graph is defined as follows:

Definition 1.1 (Simple graphs)**.** An **undirected simple graph** is a pair $G = (V(G), E(G))$, where $V(G)$ is a set, and $E(G)$ a set of unordered pairs of elements in $V(G)$:

$$E(G) \subset \{\{p, q\} \ : \ p, q \in V(G) \text{ and } p \neq q\}.$$

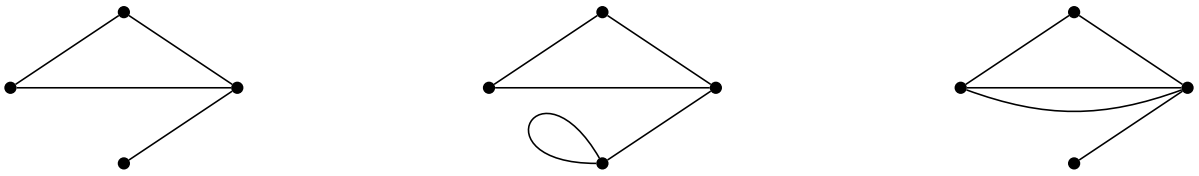
A **directed simple graph** G is constructed similarly, except that $E(G)$ consists of ordered pairs:

$$E(G) \subset \{(p, q) \ : \ p, q \in V(G) \text{ and } p \neq q\}.$$

In either case, the elements of $V(G)$ are called **vertices** and the elements of $E(G)$ are called **edges**. The **end points** of an edge are the entries of the pair.

An example of a simple graph is given in Figure 1a: the vertices are denoted as points, and edges are pictured as line segments between those vertices.

Our definition can be extended to general graphs, which can have self-loops (an edge connecting a vertex to itself, see Figure 1b) and multiple edges (more than one edge for a pair of points, see Figure 1c), but for our purposes, an intuitive idea of those properties, and thus of graphs in general, suffices.



(a) A simple graph. (b) A graph with a self-loop. (c) A graph with a double edge.

Figure 1: Graphs.

A (finite) **path** on a graph G is a subset $\{e_1, e_2, \ldots, e_n\} \subset E(G)$ such that there is a set of distinct vertices $\{p_1, p_2, \ldots, p_{n+1}\} \subset V(G)$ satisfying $e_i = \{p_i, p_{i+1}\}$; a path between $p, q \in V(G)$ is a path such that $p = p_1$ and $q = p_{n+1}$. A graph is called **connected** if there is a path between any two vertices $p, q \in V(G)$.

In our visual analogy, a graph is connected if we can draw all of its edges without lifting our pen. Thus, all graphs in Figure 1 are connected.

Definition 1.2 (Valence)**.** Let $G = (V(G), E(G))$ be an undirected graph. For a vertex $p \in V(G)$, the **valence** $v_G(p)$ is the number of edges that connect to p, where self-loops are counted twice.

We call a graph **finite** if it has finitely many vertices. Note that on a finite simple graph, the valence of any vertex is finite.

More information about graphs and related topics can be found in [11].

1.2 Metrics and measures

Let X be a set. A **metric** d on X is a non-negative, symmetric map $d : X^2 \to \mathbb{R}$ with the property that $d(x, y) = 0$ if and only if $x = y$, and which satisfies the triangle inequality

$$d(x, y) \leq d(x, z) + d(z, y)$$

for any $x, y, z \in X$. Intuitively, a metric measures the "distance" between two points in X. The pair (X, d) is called a **metric space**; when the metric d is obvious from the context, we simply refer to the metric space with X. For instance, when we mention the real line $\mathbb{R}$, we refer to the metric space $(\mathbb{R}, d)$, where $d(x, y) = |x - y|$.

We call a set $S \subset X$ **open** in (X, d) if, for every $x \in S$, there exists an $\epsilon > 0$ such that

$$\{y \in X \ : \ d(x, y) < \epsilon\} \subset S.$$

The collection of all open sets in (X, d) is called the **topology** of the metric space. A map $f \colon X \to X'$, where (X', d') is another metric space, is **continuous** if for any open set S' in (X', d'), the preimage $f^{-1}(S')$ is also open in (X, d).

A **$\boldsymbol{\sigma}$-algebra** on X is a collection Σ of subsets of X, satisfying the following three properties:

i) We have $X \in \Sigma$;

ii) If $S \in \Sigma$, then $X \setminus S \in \Sigma$;

iii) If $S_1, S_2, \ldots \in \Sigma$, then $\cup_{i=1}^{\infty} S_i \in \Sigma$.

The elements of Σ are called measurable sets. On a metric (or any topological) space (X, d), we can define a natural σ-algebra called the **Borel $\boldsymbol{\sigma}$-algebra**, denoted $\mathcal{B}(X, d)$, or $\mathcal{B}(X)$ when d is obvious. This is the smallest σ-algebra on X that contains all sets in its topology.

Given a σ-algebra Σ on X, we can define a (signed) **measure** on the measurable space (X, Σ), which is a map $\mu : \Sigma \to \mathbb{R} \cup \{-\infty, \infty\}$ that satisfies $\mu(\emptyset) = 0$ and

$$\mu\left(\bigsqcup_{i=1}^{\infty} S_i\right) = \sum_{i=1}^{\infty} \mu(S_i)$$

for any disjoint $S_1, S_2, \ldots \in \Sigma$. Informally speaking, a measure measures the "size" of the measurable sets in the corresponding σ-algebra; note that that size can be negative and infinite. A measure on a measurable space with Borel σ-algebra is also called a (signed) Borel measure.

For each measure μ, there exist measurable sets S^+ and S^- such that $X = S^+ \sqcup S^-$ and such that $\mu(S) \geq 0$ for all measurable $S \subset S^+$, and $\mu(S) \leq 0$ for all measurable $S \subset S^-$. The **variation** $|\mu|$ of μ is then defined as

$$|\mu|(S) = \mu(S \cap S^+) - \mu(S \cap S^-).$$

Given a measure μ, a measurable set S and a suitable[†] function $f : X \to \mathbb{R}$, we can integrate f over S with respect to μ. Perhaps the most famous method of integration derived this way is Lebesgue integration, using the Lebesgue measure. In situations of interest to us, Lebesgue integration is equivalent to Riemann integration. Another important measure is the Dirac measure:

Example 1.3. Let X be a metric space and $\mathcal{B}(X)$ its Borel σ-algebra. Given a fixed point $p \in X$, the Dirac measure δ_p is defined as

$$\delta_p(S) := \begin{cases} 1 & \text{if } p \in S, \\ 0 & \text{if } p \notin S. \end{cases}$$

Now, let $f\colon X \to \mathbb{R}$ and $S \in \mathcal{B}(X)$. We integrate f over S with respect to δ_p as follows:

$$\int_S f(x) d\delta_p(x) = \delta_p(S) f(p).$$

We can construct another measure by taking several $p_1, p_2, \ldots, p_n \in X$ and $c_1, c_2, \ldots, c_n \in \mathbb{R}$:

$$\mu := \sum_{i=1}^{n} c_i \delta_{p_i}.$$

The integral of f with respect to this measure is given by

$$\int_S f(x) d\mu(x) = \sum_{i=1}^{n} c_i \int_S f(x) d\delta_{p_i}(x) = \sum_{i=1}^{n} c_i \delta_{p_i}(S) f(p_i).$$

More on the theory of metric spaces can be found in [15]; more about measure and integration in [10].

[†]The function must be **integrable** with respect to the measure; all functions that we consider in this thesis are integrable, so we will not discuss this notion in detail.

2 Metrized graphs

2.1 Parametrized and metrized graphs

In Figure 1, we drew the edges of a graph as line segments (or, more generally, as curves). However, in Definition 1.1, edges are only formal links between vertices: they do not contain any points themselves, let alone a continuum like our drawings suggest. Still, it seems natural to extend the definition of a graph and regard edges as more than just abstract pairs.

This extension is formalized in the notion of a **parametrized** graph: simple graphs whose edges, as the name suggests, have parametrizations.

Definition 2.1 (Parametrized graph)**.** Let $G = (V(G), E(G))$ be a finite, undirected simple graph. Although $E(G)$ consists of undirected edges, each edge $e = \{p, q\}$ gives rise to two directed edges, called **orientations**: (p, q) and (q, p). We can refer to the edge e and one of its orientations (p, q) interchangeably by fixing (p, q) to be the orientation of e.

With each edge $e \in E(G)$, we associate a **length** $\lambda_e > 0$, and for an orientation (p, q) of e, a **parametrization** $\varphi_{(p,q)}$. That is, a bijection $\varphi_{(p,q)} \colon [0, \lambda_e] \to e$ such that $\varphi_{(p,q)}(0) = p$ and $\varphi_{(p,q)}(\lambda_e) = q$.

For the two orientations (p, q) and (q, p) of the same edge, we require that $\varphi_{(p,q)}(t) = \varphi_{(q,p)}(\lambda_e - t)$. Additionally, for two distinct edges $e, e' \in E(G)$ with fixed orientations, there is no $t \in (0, \lambda_e)$ and $t' \in (0, \lambda_{e'})$ such that $\varphi_e(t) = \varphi_{e'}(t')$.

Denote by $P(G)$ the set of parametrizations for the edges in $E(G)$. Then, we call the triple $(V(G), E(G), P(G))$ a **parametrized graph**. When clear from the context, we simply refer to it as G. The **total length** of a parametrized graph is

$$\lambda(G) := \sum_{e \in E(G)} \lambda_e.$$

Example 2.2. Consider the parametrized graph in Figure 2. Each edge has a length as indicated, and we fix their orientations as $(1, i)$, $(i, -1)$ and $(-1, 1)$. Now, we parametrize the edges as

$$\varphi_{(1,i)}(t) := e^{it}, \quad \varphi_{(i,-1)}(t) := ie^{it}, \quad \varphi_{(-1,1)}(t) := -e^{it}.$$

These maps are indeed bijective on their respective domains, and

$$\varphi_{(1,i)}(0) = 1, \quad \varphi_{(1,i)}(\tfrac{1}{2}\pi) = i;$$

$$\varphi_{(i,-1)}(0) = i, \quad \varphi_{(i,-1)}(\tfrac{1}{2}\pi) = -1;$$

$$\varphi_{(-1,1)}(0) = -1, \quad \varphi_{(-1,1)}(\pi) = 1.$$

None of the parametrizations intersect each other on the interior of their domains, either, so we have constructed a valid parametrized graph.

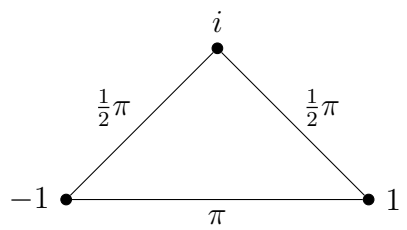


Figure 2: A parametrized graph with vertices in the complex plane.

Now that we can consider edges as curves rather than abstract links, we can look more closely at the points on an edge. Given an oriented edge e of a parametrized graph G, and a point $x = \varphi_e(t)$ for some $t \in (0, \lambda_e)$, how can we amend G so that x is also a vertex? The most natural way to do this is by **subdividing** our edge.

Figure 3: An edge and some subdivisions.

Definition 2.3 (Subdivisions of an edge)**.** Let G and G' be parametrized graphs such that $e \in E(G)$ with length λ_e, and $e'_1, e'_2, \ldots, e'_n \in E(G')$ with lengths $\lambda'_{e'_1}, \lambda'_{e'_2}, \ldots, \lambda'_{e'_n}$. We say that $\{e'_1, e'_2, \ldots, e'_n\}$ is a **subdivision** of e, written $\varsigma_{G'}(e) = \{e'_1, e'_2, \ldots, e'_n\}$, if $\lambda_e = \sum_{i=1}^n \lambda'_{e'_i}$ and we can fix orientations for $e, e'_1, \ldots, e'_n$ such that

$$\varphi_e(t) = \begin{cases} \varphi_{e'_1}(t) & \text{if } 0 \le t \le \lambda'_{e'_1}, \\ \varphi_{e'_2}\left(t - \lambda'_{e'_1}\right) & \text{if } \lambda'_{e'_1} \le t \le \lambda'_{e'_1} + \lambda'_{e'_2}, \\ & \vdots \\ \varphi_{e'_n}\left(t - \sum_{i=1}^{n-1} \lambda'_{e'_i}\right) & \text{if } \sum_{i=1}^{n-1} \lambda'_{e'_i} \le t \le \sum_{i=1}^{n} \lambda'_{e'_i}. \end{cases}$$

An edge, parametrized as a line segment, is pictured in Figure 3 along with three of its subdivisions.

Definition 2.4 (Refinements of parametrized graphs)**.** Let G and G' be parametrized graphs. We say that G' is a **refinement** of G, written $G' \le G$, if

$$E(G') = \bigcup_{e \in E(G)} \varsigma_{G'}(e).$$

Proposition 2.5 (Refinements preserve total length)**.** *Let $G' \le G$ be parametrized graphs. Then $\lambda(G') = \lambda(G)$.*

Proof. This is equivalent to saying that the union in Definition 2.4 is disjoint. Suppose that there exists an edge $e' \in E(G')$ such that $e' \in \varsigma_{G'}(e_1)$ and $e' \in \varsigma_{G'}(e_2)$, where $e_1, e_2 \in E(G)$. Then there exist $\lambda'_1 \in [0, \lambda_{e_1})$ and $\lambda'_2 \in [0, \lambda_{e_2})$[†] such that

$$\varphi_{e_1}(t) = \varphi_{e'}(t - \lambda'_1) \text{ for } \lambda'_1 \le t \le \lambda'_1 + \lambda'_{e'}$$

and

$$\varphi_{e_2}(t) = \varphi_{e'}(t - \lambda'_2) \text{ for } \lambda'_2 \le t \le \lambda'_2 + \lambda'_{e'}.$$

In particular, if we pick any $t_1 \in (\lambda'_1, \lambda'_1 + \lambda'_{e'})$ and put $t_2 = t_1 - \lambda'_1 + \lambda'_2$, then $t_2 \in (\lambda'_2, \lambda'_2 + \lambda'_{e'})$ and

$$\varphi_{e_1}(t_1) = \varphi_{e'}(t_1 - \lambda'_1) = \varphi_{e'}(t_2 - \lambda'_2) = \varphi_{e_2}(t_2).$$

By the requirements in Definition 2.1, this is not possible if $e_1 \ne e_2$. Therefore,

$$E(G') = \bigsqcup_{e \in E(G)} \varsigma_{G'}(e),$$

which implies that

$$\sum_{e' \in E(G')} \lambda'_{e'} = \sum_{e \in E(G)} \left(\sum_{e' \in \varsigma_{G'}(e)} \lambda'_{e'} \right)$$

and thus that

$$\lambda(G) = \sum_{e \in E(G)} \lambda_e = \sum_{e \in E(G)} \left(\sum_{e' \in \varsigma_{G'}(e)} \lambda'_{e'} \right) = \sum_{e' \in E(G')} \lambda'_{e'} = \lambda(G'). \qquad \square$$

[†]See Definition 2.3 for an expression for λ'_1 and λ'_2, which depend on the other edges in the respective subdivisions.

Proposition 2.6 (Refinements preserve valence)**.** *Let $G' \leq G$ be parametrized graphs. For any $p \in V(G)$, we have $p \in V(G')$ and $v_{G'}(p) = v_G(p)$.*

Proof. Subdividing an edge preserves its end points as vertices, and preserves the end points' valence. Since $E(G')$ exclusively consists of edges that form subdivisions of edges in $E(G)$, the valence of each end point of each edge in $E(G)$ is preserved. This means precisely that the valence of each vertex in $V(G)$ is preserved. □

Proposition 2.7 (Partial ordering [1, Section 2])**.** *The relationship $G' \leq G$ in Definition 2.4 defines a partial order on the set of parametrized graphs.*

Proof. We will go through the axioms of a partial order one by one.

(Reflexivity) Clearly, $E(G) = \cup_{e \in E(G)}\{e\}$, and $\varsigma_G(e) = \{e\}$ is a subdivision of e, so $G \leq G$.

(Antisymmetry) Let $G' \leq G$ and $G \leq G'$. From Proposition 2.6, we know that $G' \leq G$ implies that $V(G') \supset V(G)$, and $G \leq G'$ implies that $V(G) \supset V(G')$, so $V(G) = V(G')$. Now, considering the definition of a subdivision, the relations $G' \leq G$ and $V(G) = V(G')$ can only hold if $\varsigma_{G'}(e) = \{e\}$ for each $e \in E(G)$. This implies that $E(G') = \cup_{e \in E(G)}\{e\} = E(G)$ and thus that $G = G'$.

(Transitivity) Let $G'' \leq G'$ and $G' \leq G$. Then $E(G'')$ consists of subdivisions of the edges in $E(G')$, which in turn consists of subdivisions of edges in $E(G)$. Thus, for any edge $e \in E(G)$ we have the subdivision

$$\varsigma_{G''}(e) = \bigcup_{e' \in \varsigma_{G'}(e)} \varsigma_{G''}(e'),$$

so $G'' \leq G$. □

Not only do refinements allow us to compare parametrized graphs, they also give us an intuitive way to group parametrized graphs together, using the concept of equivalence.

Definition 2.8 (Equivalent parametrized graphs)**.** Let G and G' be two parametrized graphs. We say that G and G' are **equivalent**, written $G \sim G'$, if there exists a parametrized graph G'' such that $G'' \leq G$ and $G'' \leq G'$.

Let us check that Definition 2.8 describes an equivalence relationship.

(Reflexivity) Since refinements are a partial ordering, we have that $G \leq G$, so $G \sim G$.

(Symmetry) Let $G \sim G'$, so there is a G'' such that $G'' \leq G$ and $G'' \leq G'$. Clearly, this also implies that $G' \sim G$.

(Transitivity) Let $G \sim G'$ and $G' \sim G''$. This means that there are G_1 and G_2 such that

$$G_1 \leq G, \quad G_1 \leq G', \quad G_2 \leq G', \quad G_2 \leq G''.$$

We claim that there is a G_3 such that $G_3 \leq G_1$ and $G_3 \leq G_2$. To that end, write

$$E(G_1) = \bigcup_{e \in E(G')} \varsigma_{G_1}(e), \quad E(G_2) = \bigcup_{e \in E(G')} \varsigma_{G_2}(e).$$

We focus on a single edge $e \in E(G')$, for which we write

$$\varsigma_{G_1}(e) = \{e_1^1, e_2^1, \ldots, e_{n_1}^1\}, \quad \varsigma_{G_2}(e) = \{e_1^2, e_2^2, \ldots, e_{n_2}^2\}.$$

Each of these collections of edges gives rise to a collection of vertices by taking the end points of each edge in the subdivision:

$$\nu_{G_1}(e) := \left\{p \ : \ p \text{ is an end point of some } e^1 \in \varsigma_{G_1}(e)\right\} = \left\{p_1^1, p_2^1, \ldots, p_{n_1+1}^1\right\},$$

$$\nu_{G_2}(e) := \left\{p \ : \ p \text{ is an end point of some } e^2 \in \varsigma_{G_2}(e)\right\} = \left\{p_1^2, p_2^2, \ldots, p_{n_2+1}^2\right\}.$$

This allows us to construct a new subdivision of e (whose orientation we fix, but it does not matter which orientation we choose):

$$\varsigma_{G_3}(e) = \Big\{(p_i, p_{i+1}) \ : \ p_i, p_{i+1} \in \nu_{G_1}(e) \cup \nu_{G_2}(e) \text{ such that } \varphi_e^{-1}(p_i) < \varphi_e^{-1}(p_{i+1}) \\ \text{and there is no } p' \in \nu_{G_1}(e) \cup \nu_{G_2}(e) \text{ such that } \varphi_e^{-1}(p_i) < \varphi_e^{-1}(p') < \varphi_e^{-1}(p_{i+1})\Big\}.$$

The length of each $e^3 = (p, q) \in \varsigma_{G_3}(e)$ is given by $\lambda_{e^3} := \varphi_e^{-1}(q) - \varphi_e^{-1}(p)$, and the parametrizations are constructed to satisfy the equation in Definition 2.3, so that $\varsigma_{G_3}(e)$ is indeed a subdivision of e. Furthermore, our construction ensures that $\varsigma_{G_3}(e)$ also contains subdivisions of each $e^1 \in \varsigma_{G_1}(e)$ and each $e^2 \in \varsigma_{G_2}(e)$.

Repeating this construction for each $e \in E(G')$ yields a parametrized graph G_3 that is a refinement of both G_1 and G_2. By the transitivity property of the partial ordering, this implies that $G_3 \leq G$ and $G_3 \leq G''$, which means that $G \sim G''$.

Since all three axioms are satisfied, Definition 2.8 describes an equivalence relationship between parametrized graphs. Hence, we can group parametrized graphs with the same general structure into equivalence classes.

Definition 2.9 (Metrized graph)**.** A **metrized graph** Γ is an equivalence class of parametrized graphs. A parametrized graph G is called a **model** for the metrized graph Γ if it is in the equivalence class Γ.

Remark 2.10. The construction we used to define metrized graphs is based on the correspondence sketched in [1, Section 2], but formalizes notions such as refinements and equivalence, and defines parametrizations as a property of metrized graphs. A different but equivalent definition of a metrized graph is given in [1, Definition 2].

A metrized graph is strictly speaking not a graph itself, but we can relate it with the properties of a parametrized graph. Given a metrized graph Γ, we can choose a model G and equip Γ with the **vertex set** $V(\Gamma) := V(G)$, edge set $E(\Gamma) := E(G)$ and set of parametrizations $P(\Gamma) := P(G)$. This choice is not unique, but when we choose a vertex set, the set of edges and parametrizations uniquely follow from the corresponding model.

Proposition 2.5 tells us that the **total length** $\lambda(\Gamma) := \lambda(G)$, where G is an arbitrary model for Γ, is well-defined.

Instead of an equivalence class of graphs, we can also consider Γ as a set of points: we say that $p \in \Gamma$ if there exists a model G for Γ such that $p \in V(G)$. The **valence** $v(p)$ is then defined as $v(p) = v_G(p)$, where G is any model for Γ such that $p \in V(G)$. Proposition 2.6 ensures that this value is also well-defined.

Given a point $x \in \Gamma$, a model G for Γ and an edge $e = \{p, q\} \in E(\Gamma) = E(G)$, we say that $x \in e$ if $x = p$, $x = q$, or there is a refinement G' of G such that $\{p, x\}, \{x, q\} \in E(G')$. Note that if $x \in e$ is not an end point, we must have that $v(x) = 2$.

When we write $\Gamma - e$, we mean Γ without the interior of the edge e. That is, for $e = \{p, q\}$,

$$\{x \in \Gamma \ : \ x \notin e\} \cup \{p, q\}.$$

This notation will be used throughout this thesis.

Proposition 2.11 (Identifying points on a metrized graph)**.** *Let Γ be a metrized graph and G a model for Γ. A point x is a point of Γ if and only if there is an edge $e \in E(G)$ and a $t_x \in [0, \lambda_e]$ such that, for some orientation of e,*

$$\varphi_e(t_x) = x.$$

In particular, $x \in e$ if and only if there exists a $t_x \in [0, \lambda_e]$ such that $\varphi_e(t_x) = x$.

Proof. Let $e = (p, q) \in E(G)$ and suppose that $x \in e$. If $x = p$, then $\varphi_e(0) = x$. If $x = q$, then $\varphi_e(\lambda_e) = x$. If $x \in e \setminus \{p, q\}$, then there exists a refinement $G' \leq G$ with $e'_1 = (p, x) \in E(G')$ and $e'_2 = (x, q) \in E(G')$. By the definition of refinement, this is only possible if $\{e'_1, e'_2\}$ is a subdivision of e. Therefore,

$$\varphi_e(\lambda'_{e'_1}) = \varphi_{e'_1}(\lambda'_{e'_1}) = x.$$

Now suppose that there exists a $t_x \in [0, \lambda_e]$ such that $\varphi_e(t_x) = x$. If $t_x = 0$, then $x = p$ so $x \in e$. If $t_x = \lambda_e$, then $x = q$ so $x \in e$. Suppose that $t_x \in (0, \lambda_e)$. We can construct a subdivision $\{e'_1, e'_2\}$ of e as follows.

Put $e'_1 = (p, x)$ with length $\lambda'_{e'_1} = t_x$, and $e'_2 = (x, q)$ with $\lambda'_{e'_2} = \lambda_e - t^*$. Parametrize these edges with $\varphi_{e'_1}(t) = \varphi_e(t)$ and $\varphi_{e'_1}(t) = \varphi_e(t + t_x)$. Then $\{e'_1, e'_2\}$ is indeed a subdivision of e.

This subdivision gives rise to a refinement $G' \leq G$ with $(p, x), (x, q) \in E(G')$, so $x \in e$. □

Remark 2.12. Proposition 2.11 gives us a method to explicitly describe an abstract point $x \in \Gamma$. Let $V(\Gamma)$ be some choice of vertex set, and fix the orientation of each edge in the corresponding edge set $E(\Gamma)$. Then, $x = \varphi_e(t_x)$ for some $e \in E(\Gamma)$ and some $t_x \in [0, \lambda_e]$. In fact, if $x \notin V(\Gamma)$, this expression is unique.

We can simplify notation by denoting x by the pair (e, t_x), which are called the **coordinates** of x. If the edge we are working on is clear from the context, we may also refer to x with the **coordinate** t_x alone. Typically, by abuse of notation, we use x to denote both the abstract point of Γ and the real number t_x.

Example 2.13. Recall the parametrized graph G from Example 2.2. We want to determine the points of the metrized graph Γ that arises from G, using Proposition 2.11. To that end, we consider the union of images

$$\begin{aligned}
&\varphi_{(1,i)}\left([0, \tfrac{1}{2}\pi]\right) \cup \varphi_{(i,-1)}\left([0, \tfrac{1}{2}\pi]\right) \cup \varphi_{(-1,1)}\left([0, \pi]\right) \\
&\qquad = \left\{e^t : t \in [0, \tfrac{1}{2}\pi]\right\} \cup \left\{ie^t : t \in [0, \tfrac{1}{2}\pi]\right\} \cup \left\{-e^t : t \in [0, \pi]\right\} \\
&\qquad = \left\{e^t \ : \ t \in [0, 2\pi]\right\} \\
&\qquad = \left\{z \in \mathbb{C} \ : \ |z| = 1\right\}.
\end{aligned}$$

So Γ is the unit circle in the complex plane.

2.2 The metric on metrized graphs

From this section onward, we assume that parametrized graphs are connected, and thus that models of a metrized graph are connected. Given a vertex set $V(\Gamma)$ for a metrized graph Γ, we say that an edge $e = \{p, q\} \in E(\Gamma)$ is a **bridge** if the models for Γ are connected, but the models for $\Gamma - e$ are not. In that case, the models for $\Gamma - e$ consist of two disconnected subgraphs, which in turn give rise to two metrized graphs: Γ_p containing p, and Γ_q containing q. We will use this notation throughout the rest of this thesis.

Example 2.14. Consider the metrized graph Γ for which a model is shown in Figure 4. We use this model to equip Γ with the vertex set $V(\Gamma)$.

The edge $e \in E(\Gamma)$ is a bridge: if e were deleted, we would be left with a subgraph with vertices $\{p_1, p_2, p_3\}$, and a subgraph with vertices $\{p_4, p_5, p_6\}$, without an edge between the two subgraphs. These subgraphs are models for the metrized graphs Γ_{p_3} and Γ_{p_4}. Note that $p_1, p_2, p_3 \in \Gamma_{p_3}$ and $p_4, p_5, p_6 \in \Gamma_{p_4}$.

In general, for a bridge $e = \{p, q\}$ and an edge $e' \neq e$, either $e' \subset \Gamma_p$ or $e' \subset \Gamma_q$.

With our assumption that parametrized graphs are connected, we can define a metric on parametrized graph, which also allows us to define a metric on metrized graphs.

Definition 2.15 (Metric on a parametrized graph)**.** Let G be a parametrized graph and $p, q \in V(G)$. The metric $d_G(p, q)$ is defined as the length of the shortest path between p and q. That is,

$$d_G(p,q) := \min\left\{\sum_{i=1}^{n} \lambda_{e_i} \ : \ \{e_1, e_2, \ldots, e_n\} \text{ is a path between } p \text{ and } q\right\}.$$

Remark 2.16. Note that d_G only relies on the lengths of the edges of G, not the parametrizations of those edges. In fact, the metric in Definition 2.15 describes a specific case of the **graph distance**. This quantity is thoroughly studied in graph theory and known to be a metric on connected undirected graphs whose edges have lengths [11, Chapter 2, "Walks and connectedness"]. So, in particular, d_G is indeed a metric in the parametrized graph G.

Definition 2.17 (Metric on a metrized graph)**.** Let Γ be a metrized graph and $p, q \in \Gamma$. The metric $d(p, q)$ is defined as

$$d(p,q) := d_G(p,q),$$

where G is any model for Γ such that $p, q \in V(G)$.

Lemma 2.18 (Well-definedness of the metric)**.** *Let Γ be a metrized graph with $p, q \in \Gamma$. The value of $d(p, q)$ as defined in 2.17 is independent of our choice of a model G satisfying $p, q \in V(G)$.*

Proof. Let G and G' be models for Γ such that $p, q \in V(G)$ and $p, q \in V(G')$. Since $G \sim G'$, there exists a model G'' such that $G'' \leq G$ and $G'' \leq G'$.

Let $\{e_1, e_2, \ldots, e_n\} \subset E(G)$ be a path between p and q such that $d_G(p, q) = \sum_{i=1}^{n} \lambda_{e_i}$. Then there is a path

$$\bigcup_{i=1}^{n} \varsigma_{G''}(e_i) \subset E(G'')$$

between p and q with, by the same reasoning we used to prove Proposition 2.5, length

$$\sum_{i=1}^{n}\left(\sum_{e'' \in \varsigma_{G''}(e_i)} \lambda''_{e''}\right) = \sum_{i=1}^{n} \lambda_{e_i} = d_G(p,q).$$

Thus, the length of the shortest path on G'' between p and q is at most $d_G(p, q)$:

$$d_{G''}(p,q) \leq d_G(p,q).$$

Now, let $\omega = \{e''_1, e''_2, \ldots, e''_m\} \subset E(G'')$ be a path between p and q such that $d_{G''}(p, q) = \sum_{i=1}^{m} \lambda''_{e''_i}$. Let π denote the set of end points of the edges in ω:

$$\pi = \{p''_1 = p, p''_2, \ldots, p''_{m+1} = q\} \subset V(G'').$$

Intersecting this set with $V(G)$ yields a set of vertices of G, which includes p and q:

$$\pi \cap V(G) = \{p''_{k_1} = p, p''_{k_2}, \ldots, p''_{k_{n^*+1}} = q\}.$$

Put $e^*_i = \{p''_{k_i}, p''_{k_{i+1}}\}$ and consider the following path between p''_{k_i} and $p''_{k_{i+1}}$:

$$\left\{\{p''_{k_i}, p''_{k_i+1}\}, \{p''_{k_i+1}, p''_{k_1+2}\}, \ldots, \{p''_{k_{i+1}-1}, p''_{k_{i+1}}\}\right\} \subset E(G'') \tag{2.18.1}$$

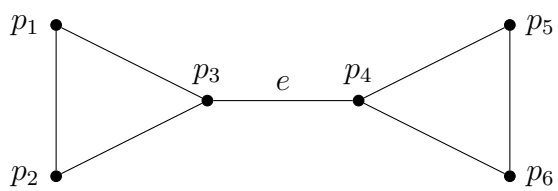


Figure 4: A model for our metrized graph.

Only two end points of these edges are also vertices of G: p''_{k_i} and $p''_{k_{i+1}}$. Hence, since G'' is a refinement of G, we have $e_i^* \in E(G)$, and the path in 2.18.1 is the subdivision $\varsigma_{G''}(e_i^*)$.

This holds for all e_i^*, so

$$\omega = \bigcup_{i=1}^{n^*} \varsigma_{G''}(e_i^*).$$

Therefore,

$$d_{G''}(p,q) = \sum_{i=1}^{m} \lambda''_{e''_i} = \sum_{i=1}^{n^*} \left(\sum_{e'' \in \varsigma_{G''}(e_i^*)} \lambda''_{e''} \right) = \sum_{i=1}^{n^*} \lambda_{e_i^*} \geq d_G(p,q).$$

This yields $d_G(p,q) = d_{G''}(p,q)$ and, by the same reasoning, $d_{G'}(p,q) = d_{G''}(p,q)$. Therefore, we have that $d_G(p,q) = d_{G'}(p,q)$ for any models G and G' with vertices p and q. □

Remark 2.19. Since the graph distance is a metric on parametrized graphs, it follows that d from Definition 2.17 is also a metric. Indeed, let Γ be a metrized graph, $x, y, z \in \Gamma$, and G a model for Γ such that $x, y, z \in V(G)$. Then $d(x,y) = d_G(x,y) \geq 0$, with $d(x,y) = 0$ only if $x = y$. Furthermore, $d(x,y) = d_G(x,y) = d_G(y,x) = d(y,x)$, and the triangle inequality holds:

$$d(x,y) = d_G(x,y) \leq d_G(x,z) + d_G(z,y) = d(x,z) + d(z,y).$$

So Definition 2.17 indeed defines a metric on a metrized graph. Additionally, notice that d is independent of the parametrizations, similarly to d_G.

With Definition 2.17 in mind, we can regard a metrized graph as a metric space. From this point onward, when we mention a metrized graph Γ, we refer to the metric space (Γ, d), equipped with the metric d from Definition 2.17.

Given a map between a metrized graph and another metric space, we can now make sense of the notion of continuity.

If two distinct edges $e, e' \in E(\Gamma)$ are bridges, the **closest neighbours** of e and e' are defined as the points on e respectively e' that are a minimal distance from each other, i.e. the entries of the unique pair

$$\langle e, e' \rangle := (x, x')$$

for which

$$d(x,x') = \min \{ d(y,y') \ : \ y \in e \text{ and } y' \in e' \}.$$

The closest neighbours of a pair of edges are end points of the respective edges. Specifically, when $e = (p,q)$ and $e' = (p',q')$, then there are four options for closest neighbours:

$$\langle e, e' \rangle \subset \{ (p,p'),\ (p,q'),\ (q,p),\ (q,q') \}.$$

This terminology and observation will be used later, too.

Example 2.20. Consider the metrized graph with the vertex set of the model in Figure 5. The edges e and e' are bridges, and the closest neighbours are given by

$$\langle e, e' \rangle = (p_2, p_5).$$

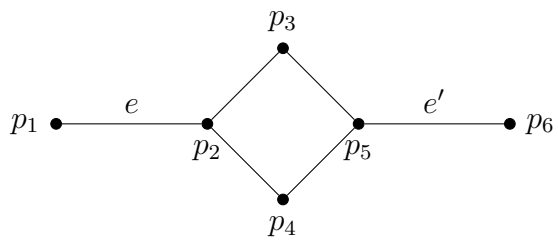


Figure 5: A model for a metrized graph with two bridges.

Note that we do not need to know the length of the edges to determine the closest neighbours: a path between, for instance, p_1 and p_5 will always be longer than the shortest path between p_2 and p_5, because the former path will always include e.

2.3 Measures on metrized graphs

Now that we consider metrized graphs as metric spaces, we can equip the metrized graph Γ with the Borel σ-algebra $\mathcal{B}(\Gamma)$. On the resulting measurable space $(\Gamma, \mathcal{B}(\Gamma))$, we can define Borel measures. We are particularly interested in measures of the form in Lemma 2.21:

Lemma 2.21 (Borel measures on metrized graphs)**.** *Let Γ be a metrized graph equipped with vertex set $V(\Gamma)$, and let $n \in \mathbb{Z}$. For each edge e, let $g_e \colon \Gamma \to \mathbb{R}$ be a continuous and bounded function, and, for each $i \in \{1, 2, \ldots, n\}$, let $c_i \in \mathbb{R}$. Then*

$$\mu = \sum_{e \in E(\Gamma)} g_e(x) dx|_e + \sum_{i=1}^{n} c_i \delta_{p_i}, \tag{2.21.1}$$

where $dx|_e$ denotes the Lebesgue measure on e, and δ_p the Dirac measure on Γ, is a Borel measure on Γ.

Proof. Let $e \in E(\Gamma)$. Since e is closed in Γ, e is in $\mathcal{B}(\Gamma)$. The Lebesgue measure dx is defined on the Borel σ-algebra of e, and the topology of e is given by

$$\{\text{open sets of } e\} = \{S \cap e \ : \ S \text{ is open in } \Gamma\},$$

which implies that

$$\mathcal{B}(e) = \{S \cap e \ : \ S \in \mathcal{B}(e)\}.$$

Therefore, the measure

$$dx|_e(S) := dx(S \cap e)$$

is defined on $\mathcal{B}(\Gamma)$. Scaling $dx|_e$ with a real-valued continuous and bounded function does not change its domain of definition.

As we saw in Example 1.3, the Dirac measure is a Borel measure, so δ_p for $p \in \Gamma$ is a Borel measure on Γ. Scaling δ_p with a real constant does not change its domain of definition.

Thus, all terms in 2.21.1 are Borel measures. Summing finitely many Borel measures again yields a Borel measure, so the measure in 2.21.1 is a Borel measure on Γ. □

For a function $f : \Gamma \to \mathbb{R}$ and an edge $e \in E(\Gamma)$, we integrate as follows:

$$\int_e f(x) dx|_e = \int_0^{\lambda_e} f(\varphi_e(t)) dt \quad \text{and} \quad \int_\Gamma f(x) d\delta_p(x) = f(p).$$

When integrating over the entire metrized graph with respect to the Lebesgue measure, we write

$$\int_\Gamma f(x) dx := \sum_{e \in E(\Gamma)} \int_e f(x) dx|_e.$$

Thus, for a measure of the general form in Equation 2.21.1,

$$\int_\Gamma f(x) d\mu(x) = \sum_{e \in E(\Gamma)} \int_0^{\lambda_e} f(\varphi_e(t)) g_e(\varphi_e(t)) dt + \sum_{i=1}^{n} c_i f(p_i).$$

To compute this integral, we need to fix an orientation for each edge, but the value is independent of our choice.

2.4 The discrete Laplacian matrix

Definition 2.22 (Discrete Laplacian matrix [8, Section 2])**.** Let Γ be a metrized graph with vertex set $V(\Gamma) = \{p_1, p_2, \ldots, p_n\}$. Then the **discrete Laplacian matrix** of Γ (for this vertex set) is an $n \times n$ matrix given by $L := (l_{ij})$, where

$$l_{ij} := \begin{cases} 0 & \text{if } i \neq j \text{ and } \{p_i, p_j\} \notin E(\Gamma), \\ -\frac{1}{\lambda_{\{p_i,p_j\}}} & \text{if } i \neq j \text{ and } \{p_i, p_j\} \in E(\Gamma), \\ -\sum_{k \in \{1,2,\ldots,n\} \setminus \{i\}} l_{ik} & \text{if } i = j. \end{cases}$$

The discrete Laplacian matrix need not be invertible, so L does not necessarily have an inverse. However, it does have a pseudo-inverse L^+:

Definition 2.23 (Moore–Penrose pseudo-inverse [7, Section 3])**.** Let $M \in \mathbb{R}^{m \times n}$. The Moore–Penrose **pseudo-inverse** is defined as the matrix $M^+ \in \mathbb{R}^{n \times m}$ satisfying all of the following conditions:

i) $MM^+M = M$;

ii) $M^+MM^+ = M^+$;

iii) $(MM^+)^T = MM^+$;

iv) $(M^+M)^T = M^+M$.

It can be shown that M^+ exists and is unique for any M, so in particular, for any discrete Laplacian matrix L.

Proposition 2.24 (Properties of the discrete Laplacian matrix and its pseudo-inverse [14, Theorem 10.1.2])**.** *Let Γ be a metrized graph with equipped with $V(\Gamma) = \{p_1, p_2, \ldots, p_n\}$, and let L be the corresponding discrete Laplacian matrix. Then*

$$L^+ = \left(L - \tfrac{1}{n}J\right)^{-1} + \tfrac{1}{n}J,$$

where J is the $n \times n$ matrix with all entries 1. In particular, since L is symmetric, L^+ is also symmetric.

2.5 Polarized metrized graphs

We can extend the definition of a metrized graph by assigning an integer to each point, yielding a **polarized metrized graph**:

Definition 2.25 (Polarized metrized graph)**.** Let Γ be a metrized graph and let $\rho : \Gamma \to \mathbb{Z}_{\geq 0}$ be a map satisfying $\rho(p) \neq 0$ for finitely many $p \in \Gamma$. We then call ρ a **polarization** and the pair (Γ, ρ) a **polarized metrized graph**.

Example 2.26. Recall the complex unit circle Γ from Example 2.13. Define the polarization ρ as $\rho(1) = 1$, $\rho(i) = 2$, and $\rho(x) = 0$ for all $x \in \Gamma \setminus \{1, i\}$. Then (Γ, ρ) is a polarized metrized graph.

Polarized metrized graphs arise from semistable models for algebraic curves, as discussed in [9] and [12]. This theory is beyond the scope of this thesis, but we will use polarized metrized graphs later to define the canonical divisor in Definition 5.12.

3 Piecewise smooth functions on a metrized graph

3.1 Derivatives on metrized graphs

Metrized graphs combine the discrete properties of graphs with the continuous properties of curves. As such, we can consider notions such as differentiability and smoothness of functions on metrized graphs, albeit with some peculiarities.

This section summarizes the most important definitions and results in [1, Section 3].

Definition 3.1 (Derivative at a vertex)**.** Let Γ be a metrized graph with vertex set $V(\Gamma)$ and $f : \Gamma \to \mathbb{R}$. Given an edge $e = (p, q) \in E(\Gamma)$, we define the **directional derivatives** of the vertices p and q along e as

$$D_e f(p) := \lim_{\epsilon \to 0^+} \frac{f(\varphi_e(\epsilon)) - f(p)}{\epsilon},$$
$$D_e f(q) := \lim_{\epsilon \to 0^+} \frac{f(q) - f(\varphi_e(\lambda_e - \epsilon))}{\epsilon},$$

provided those limits exist.

Note that, if we explicitly write the orientations, $D_{(p,q)} f(p) = -D_{(q,p)} f(p)$. If $D_e f(p)$ exists, we say that f is **differentiable** at p along e. We can also define differentiability on points that are not vertices:

Definition 3.2 (Differentiability on non-vertices)**.** Let Γ be a metrized graph with vertex set $V(\Gamma)$, $e \in E(\Gamma)$ and $f \colon \Gamma \to \mathbb{R}$. We say that f is ***n*-times differentiable** at $x \in e \setminus V(\Gamma)$ if, for some orientation of e,

$$\frac{d^n}{dt^n} f(\varphi_e(t)) \bigg|_{\varphi_e(t)=x}$$

exists.

Although the existence of this quantity for one orientation directly implies that it exists for the other orientation, the values need not be equal for both orientations. In fact, by the chain rule,

$$\frac{d}{dt} f(\varphi_{(p,q)}(t)) \bigg|_{\varphi_{(p,q)}(t)=x} = -\frac{d}{dt} f(\varphi_{(q,p)}(t)) \bigg|_{\varphi_{(q,p)}(t)=x}.$$

However, the second derivative on the interior of an edge is independent of the chosen orientation:

$$\begin{aligned}
\frac{d^2}{dt^2} f(\varphi_{(p,q)}(t)) \bigg|_{\varphi_{(p,q)}(t)=x} &= \frac{d}{dt} \left[\frac{d}{dt} \left[f \circ \varphi_{(p,q)} \right](t) \right] \bigg|_{\varphi_{(p,q)}(t)=x} \\
&= \frac{d}{dt} \left[-\frac{d}{dt} \left[f \circ \varphi_{(q,p)} \right](\lambda_{(p,q)} - t) \right] \bigg|_{\varphi_{(q,p)}(\lambda_{(p,q)}-t)=x} \\
&= -\frac{d}{dt} \left[\frac{d}{dt} \left[f \circ \varphi_{(q,p)} \right](\lambda_{(p,q)} - t) \right] \bigg|_{\varphi_{(q,p)}(\lambda_{(p,q)}-t)=x} \\
&= \frac{d^2}{dt^2} \left[f \circ \varphi_{(q,p)} \right](\lambda_{(p,q)} - t) \bigg|_{\varphi_{(q,p)}(\lambda_{(p,q)}-t)=x} \\
&= \frac{d^2}{dt^2} \left[f \circ \varphi_{(q,p)} \right](t) \bigg|_{\varphi_{(q,p)}(t)=x} \\
&= \frac{d^2}{dt^2} f(\varphi_{(q,p)}(t)) \bigg|_{\varphi_{(q,p)}(t)=x}.
\end{aligned}$$

We can thus safely define the second derivative on the interior of an edge.

Definition 3.3 (Second derivative)**.** Let Γ be a metrized graph with vertex set $V(\Gamma)$, $e \in E(\Gamma)$, and let $f \colon \Gamma \to \mathbb{R}$ be twice differentiable at $x \in e \setminus V(\Gamma)$. Then the **second derivative** of f at x is defined as

$$f''(x) := \frac{d^2}{dt^2} f(\varphi_e(t)) \bigg|_{\varphi_e(t)=x}.$$

We can now classify especially "nice" functions on metrized graphs, which we call **piecewise smooth**. We restrict ourselves to these functions in the rest of this text.

Definition 3.4 (Piecewise smooth functions)**.** Let Γ be a metrized graph and let $f\colon \Gamma \to \mathbb{R}$ be continuous. We say that f is **piecewise smooth** if there exists a model G_f for Γ such that all of the following conditions are satisfied:

i) For each $p \in V(G_f)$ and each $e \in E(G_f)$ such that $p \in e$, we have that $D_e f(p)$ exists;

ii) For each $e \in E(G_f)$, f is twice differentiable at every $x \in e \setminus V(G_f)$;

iii) For each $e \in E(G_f)$, f'' is bounded on $e \setminus V(G_f)$.

Remark 3.5. Some readers may object that Definition 3.4 only requires twice-differentiability, while smooth functions from calculus are infinitely differentiable. The name "piecewise twice-differentiable" would thus be more appropriate, but that term is awkwardly long, and for our purposes, twice-differentiable functions are *smooth enough*.

3.2 The Laplacian operator

Being piecewise smooth on a metrized graph is similar to being twice-differentiable on $\mathbb{R}^n$. As such, it makes sense to define an analogy for the real Laplacian operator on metrized graphs. Unlike its Euclidean counterpart however, the Laplacian of a piecewise smooth function on a metrized graph is not a function, but a measure of the form in 2.21.1.

Definition 3.6 (Laplacian operator)**.** Let Γ be a metrized graph and let $f\colon \Gamma \to \mathbb{R}$ be piecewise smooth, with G_f a model as described in Definition 3.4. The **Laplacian** of f is defined as the following signed Borel measure on Γ:

$$\Delta f := \sum_{e \in E(G_f)} \left[-f''(x)dx|_e + D_e f(\varphi_e(\lambda_e))\delta_{\varphi_e(\lambda_e)} - D_e f(\varphi_e(0))\delta_{\varphi_e(0)} \right].$$

Remark 3.7. Definition 3.6 differs from [1, Definition 5], but in the same text, the two definitions were shown to be equivalent. We opted to define the Laplacian operator this way because it allows for easier computations and because it coincides better with the notation we have used thus far. For instance, we can immediately tell that the Laplacian of a function is a Borel measure because it is of the form in 2.21.1.

Remark 3.8. We need to choose a model G_f and an orientation for each edge for Definition 3.6 to make sense, but the resulting measure is independent of our choice. This can be proven using the fact that if a model G_f satisfies the properties in Definition 3.4, then a refinement $G'_f \leq G_f$ is satisfactory too. Because

$$f(x)dx|_e = \sum_{e' \in \varsigma_{G'_f}(e)} f(x)dx|_e$$

for each $e \in V(G_f)$, and via the same reasoning we used in the proof of Lemma 2.18, we find that the Laplacian is independent of our choice of model. Independence of orientation follows from our earlier discovery that $D_{(p,q)}f(p) = -D_{(q,p)}f(p)$.

Example 3.9. Consider the complex unit circle Γ, that is, the metrized graph from Example 2.13. Define $f\colon \Gamma \to \mathbb{R}$ by

$$f(z) := \begin{cases} \mathrm{Re}(z) & \text{if } 0 \leq \mathrm{Arg}(z) \leq \frac{1}{2}\pi, \\ \mathrm{Im}(z) - 1 & \text{if } \frac{1}{2}\pi \leq \mathrm{Arg}(z) \leq \frac{3}{2}\pi, \\ \mathrm{Re}(z^2) + \mathrm{Im}(z) & \text{if } \frac{3}{2}\pi \leq \mathrm{Arg}(z) \leq 2\pi. \end{cases}$$

A suitable G_f is as pictured in Figure 6, with edges enumerated and oriented as indicated. The corresponding parametrizations are

$$\varphi_{e_1}(t) = e^{it}, \quad t \in [0, \tfrac{1}{2}\pi];$$

$$\varphi_{e_2}(t) = ie^{it}, \quad t \in [0, \pi];$$

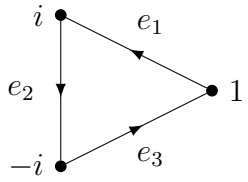


Figure 6: Parametrized graph G_f for the function in example 3.9.

$$\varphi_{e_3}(t) = -ie^{it}, \quad t \in [0, \tfrac{1}{2}\pi].$$

We will compute the Laplacian using

$$\Delta f = \sum_{k=1}^{3} \left[-f''(z)dz|_{e_k} + D_{e_k} f(\varphi_{e_k}(\lambda_{e_k}))\delta_{\varphi_{e_k}(\lambda_{e_k})} - D_{e_k} f(\varphi_{e_k}(0))\delta_{\varphi_{e_k}(0)} \right]. \tag{3.9.1}$$

We start with $k = 1$. On e_1, we have

$$\frac{d^2}{dt^2} f(\varphi_{e_1}(t)) = \frac{d^2}{dt^2} \operatorname{Re}(e^{it}) = \frac{d}{dt^2} \cos(t) = -\cos(t).$$

Combining $\varphi_{e_1}(t) = z$ with the fact that $\operatorname{Re}(e^{it}) = \cos(t)$ gives us

$$f''(z) = -\operatorname{Re}(z)$$

on the interior of e_1. As for the vertices,

$$D_{e_1} f(\varphi_{e_1}(0)) = \lim_{\epsilon \to 0^+} \frac{f(\varphi_{e_1}(\epsilon)) - f(\varphi_{e_1}(0))}{\epsilon} = \lim_{\epsilon \to 0^+} \frac{\cos(\epsilon) - \cos(0)}{\epsilon} = \frac{d}{dt} \cos(t) \bigg|_{t=0} = -\sin(0) = 0$$

and similarly

$$D_{e_1} f(\varphi_{e_1}(\tfrac{1}{2}\pi)) = \sin(\tfrac{1}{2}\pi) = 1.$$

Thus the first summand of 3.9.1 is

$$\operatorname{Re}(z)dz|_{e_1} + \delta_i. \tag{3.9.2}$$

The other terms can be computed in a similar fashion:

$$\operatorname{Im}(z)dz|_{e_2}, \tag{3.9.3}$$

and

$$(4\operatorname{Re}(z^2) + \operatorname{Im}(z))dz|_{e_3} - \delta_1. \tag{3.9.4}$$

Summing equations 3.9.2, 3.9.3 and 3.9.4 yields

$$\Delta f = \operatorname{Re}(z)dz|_{e_1} + \operatorname{Im}(z)dz|_{e_2} + \left(4\operatorname{Re}(z^2) + \operatorname{Im}(z)\right) dz|_{e_3} - \delta_1 + \delta_i.$$

The Laplacian can be generalized for multivariate functions, e.g. $f\colon \Gamma^2 \to \mathbb{R}$ with variables x and y:

$$\Delta_x f(y) := \sum_{e \in E(G_f)} \left[-f''(x,y)dx|_e + D_e f(\varphi_e(\lambda_e), y)\delta_{\varphi_e(\lambda_e)} - D_e f(\varphi_e(0), y)\delta_{\varphi_e(0)} \right],$$

where we consider y as a constant and the derivatives are taken with respect to x. The object $\Delta_x f$ is thus a map from Γ to the set of Borel measures on Γ.

Further discussion of the Laplacian, as well as more computational examples, can be found in [1] and [2].

4 Voltage and resistance functions

4.1 Metrized graphs as electrical networks

In this section, we discuss the correspondence between metrized graphs and electrical circuits. The basis of this interpretation is the **voltage function** j, whose values can be interpreted as follows. If a model G of Γ with $x, y, z \in V(G)$ is viewed as an electrical network, with the resistance of each connection (edge) given by its length, then $j_z(x, y)$ is the voltage difference between x and z when unit current enters at y and exits at z, with reference voltage 0 at z; see [8, Section 3].

More rigorously, the voltage function is defined using the Laplacian.

Theorem 4.1 (Voltage function [1, Corollary 3 and Exercise 8])**.** *Let Γ be a metrized graph. There exists a unique function $j\colon \Gamma^3 \to \mathbb{R}$ such that*

$$\Delta_x j_z(x, y) = \delta_y - \delta_z \quad \text{and} \quad j_z(z, y) = 0.$$

*This function j is called the **voltage function** on Γ.*

We also define the **resistance function**, which corresponds to the physical concept of effective resistance between two points on an electrical circuit.

Definition 4.2 (Resistance function)**.** Let Γ be a metrized graph. The **resistance function** r is defined as $r(x, y) := j_y(x, x)$.

With the physical interpretation in mind, the following results should not be surprising:

Proposition 4.3 (Properties of the voltage and resistance function [8, Section 3])**.** *Let Γ be a metrized graph, let $w, x, y, z \in \Gamma$ and let j and r be the voltage and resistance function on Γ, respectively. Then $j_z(x, y) \geq 0$ and $j_x(x, y) = 0$, so $r(x, y) \geq 0$ and $r(x, x) = 0$. Furthermore, we have that*

$$j_z(x, y) - j_z(w, y) = j_w(y, x) - j_w(z, x),$$

$j_z(x, y) = j_z(y, x)$ and $j_z(x, x) = j_x(z, z)$. Therefore, $r(x, y) = r(y, x)$.

Lemma 4.4 (Expression of j in terms of r [8, Remark 3.5])**.** *Let Γ be a metrized graph and $x, y, z \in \Gamma$. Then*

$$j_z(x, y) = \tfrac{1}{2}\left(r(x, z) + r(y, z) - r(x, y)\right).$$

Remark 4.5. Although the physical interpretation of j and r require us to choose a vertex set (and thus the edges, i.e. wires), their formal definition ensures us that they are independent of our choice. Furthermore, the function values only depend on the lengths of edges, not on their parametrizations. Physically, this means that voltage differences and effective resistances depend on the resistances of the wires, not on the shapes in which they are laid out.

4.2 Explicit formulas for the resistance function

Although Proposition 4.3 confirms the physical interpretation of the voltage and resistance functions, we have yet to find a method to easily compute their values; from Example 3.9 we know that directly working with the Laplacian can be rather cumbersome.

It turns out that, if we fix a vertex set for our metrized graph, we can compute the function values at the vertices using the discrete Laplacian matrix:

Theorem 4.6 (Resistance and voltage function at vertices [6, Lemmas 3.4 and 3.5])**.** *Let Γ be a metrized graph with $V(\Gamma) = \{p_1, p_2, \ldots, p_n\}$. Denote the corresponding discrete Laplacian matrix by $L = (l_{ij})$ and its pseudo-inverse by $L^+ = (l^+_{ij})$. Then,*

$$r(p_i, p_j) = l^+_{ii} - 2l^+_{ij} + l^+_{jj} \quad \text{and} \quad j_{p_k}(p_i, p_j) = l^+_{kk} - l^+_{ik} - l^+_{jk} + l^+_{ij}.$$

Proof. This can be proven by interpreting Γ as an electrical circuit and $r(p_i, p_j)$ as the effective resistance between the nodes p_i and p_j. The result then follows from Ohm's law and Kirchhoff's voltage law [3, Chapter 10]. □

Lemma 4.7 gives formulas for the resistance function if one of the points is not a vertex; this is a special case of Theorem 4.8 and will be used later. Note that, because of Lemma 4.4, this theorem also gives us an easy method to compute values of the voltage function.

Recall the following conventions. If we equip a metrized graph Γ with the vertex set $V(\Gamma)$, and fix the orientation of an edge e in the corresponding edge set $E(\Gamma)$, then we can refer to the edge by that orientation. As discussed in Remark 2.12, for a point $x \in e$, we also denote the coordinate t_x, for which $x = \varphi_e(t_x)$, by x.

When $e = (p, q)$ is a bridge, the connected metrized graphs that arise when e is deleted from the model are denoted by Γ_p, which contains p, and Γ_q, which contains q. If $e' \in E(G)$ is a different bridge, then $\langle e, e' \rangle$ denotes the pair of closest neighbours of e and e' respectively.

Lemma 4.7 (Formulas for the resistance function at one vertex [8, Lemma 3.2 and Remark 3.4.2])**.** *Let Γ be a metrized graph with vertex set $V(\Gamma)$ and edges with fixed orientations. Let $(p, q) \in E(\Gamma)$, $x \in (p, q)$, and $s \in V(\Gamma)$. If (p, q) is not a bridge, then*

$$r(s, x) = -x^2 \frac{\lambda_{(p,q)} - r(p, q)}{\lambda_{(p,q)}^2} + x \frac{\lambda_{(p,q)} - r(p, q) + r(s, q) - r(s, p)}{\lambda_{(p,q)}} + r(s, p). \tag{4.7.1}$$

If (p, q) is a bridge, then

$$r(s, x) = \begin{cases} x + r(s, p) & \text{if } s \in \Gamma_p, \\ \lambda_{(p,q)} - x + r(s, q) & \text{if } s \in \Gamma_q. \end{cases} \tag{4.7.2}$$

Proof. A proof of this result can be found in [8], which again exploits the correspondence between r and the effective resistance on an electric circuit: relations between serial and parallel resistors allow for "circuit reductions" that yield a graph that is simpler but equivalent in terms of resistance. More details on this technique can be found in [16]. □

Theorem 4.8 (Formulas for the resistance function [8, Lemma 3.1, Theorem 3.3 and Remark 3.4])**.** *Let Γ be a metrized graph with vertex set $V(\Gamma)$ and edges with fixed orientations. Let $(p, q) \in E(\Gamma)$ and $x, y \in (p, q)$. If (p, q) is not a bridge, then*

$$r(x, y) = |x - y| - (x - y)^2 \frac{\lambda_{(p,q)} - r(p, q)}{\lambda_{(p,q)}^2}. \tag{4.8.1}$$

If (p, q) is a bridge, then

$$r(x, y) = |x - y|. \tag{4.8.2}$$

Now let $(p_1, q_1), (p_2, q_2) \in E(\Gamma)$ such that $\{p_1, q_1\} \neq \{p_2, q_2\}$, so that the edges are distinct but may share an end point, and let $x \in (p_1, q_1)$ and $y \in (p_2, q_2)$. If neither edge is a bridge, then

$$\begin{aligned} r(x, y) = &- x^2 \frac{\lambda_{(p_1,q_1)} - r(p_1, q_1)}{\lambda_{(p_1,q_1)}^2} - y^2 \frac{\lambda_{(p_2,q_2)} - r(p_2, q_2)}{\lambda_{(p_2,q_2)}^2} + \frac{2xy}{\lambda_{(p_1,q_1)} \lambda_{(p_2,q_2)}} \left(j_{p_2}(p_1, q_2) - j_{p_2}(q_1, q_2) \right) \\ &+ \frac{x}{\lambda_{(p_1,q_1)}} \left(\lambda_{(p_1,q_1)} - 2 j_{p_1}(q_1, p_2) \right) + \frac{y}{\lambda_{(p_2,q_2)}} \left(\lambda_{(p_2,q_2)} - 2 j_{p_2}(p_1, q_2) \right) + r(p_1, p_2). \end{aligned} \tag{4.8.3}$$

If (p_1, q_1) is a bridge but (p_2, q_2) is not, then

$$r(x, y) = \begin{cases} x - y^2 \dfrac{\lambda_{(p_2,q_2)} - r(p_2, q_2)}{\lambda_{(p_2,q_2)}^2} \\ \quad + y \dfrac{\lambda_{(p_2,q_2)} - r(p_2, q_2) + r(p_1, q_2) - r(p_1, p_2)}{\lambda_{(p_2,q_2)}} + r(p_1, p_2) & \text{if } (p_2, q_2) \subset \Gamma_{p_1}, \\ \\ \lambda_{(p_1,q_1)} - x - y^2 \dfrac{\lambda_{(p_2,q_2)} - r(p_2, q_2)}{\lambda_{(p_2,q_2)}^2} \\ \quad + y \dfrac{\lambda_{(p_2,q_2)} - r(p_2, q_2) + r(q_1, q_2) - r(q_1, p_2)}{\lambda_{(p_2,q_2)}} + r(q_1, p_2) & \text{if } (p_2, q_2) \subset \Gamma_{q_1}. \end{cases} \tag{4.8.4}$$

Lastly, if both (p_1, q_1) *and* (p_2, q_2) *are bridges, then*

$$r(x,y) = \begin{cases} x + y + r(p_1, p_2) & \text{if } \langle (p_1, q_1), (p_2, q_2) \rangle = (p_1, q_1), \\ x + \lambda_{(p_2,q_2)} - y + r(p_1, q_2) & \text{if } \langle (p_1, q_1), (p_2, q_2) \rangle = (p_1, q_2), \\ \lambda_{(p_1,q_1)} - x + y + r(q_1, p_2) & \text{if } \langle (p_1, q_1), (p_2, q_2) \rangle = (q_1, p_2), \\ \lambda_{(p_1,q_1)} - x + \lambda_{(p_2,q_2)} - y + r(q_1, q_2) & \text{if } \langle (p_1, q_1), (p_2, q_2) \rangle = (q_1, q_2). \end{cases} \tag{4.8.5}$$

Proof. Each of these formulas can be proven similarly to Lemma 4.7. □

5 Arakelov–Green functions

5.1 Arakelov–Green functions

Originally introduced by Ted Chinburg, Robert Rumely and Shouwu Zhang, and inspired by the Green's functions that Suren Arakelov used on Riemann surfaces, **Arakelov–Green functions** are a class of real-valued functions on metrized graphs that have important applications in number theory [4]. Furthermore, several important invariants on metrized graphs are defined using Arakelov–Green functions [9]. We discuss one of these invariants in Section 7.4.

The details of this motivation are beyond the scope of this thesis. Instead, this section defines Arakelov–Green functions and considers specific cases with respect to the canonical measure and other admissible metrics.

Definition 5.1 (Arakelov–Green function)**.** Let Γ be a metrized graph and let μ be a signed Borel measure on Γ such that $\mu(\Gamma) = 1$ and $|\mu|(\Gamma) < \infty$. The **Arakelov–Green function** g_μ is defined as

$$g_\mu(x, y) := \int_\Gamma j_z(x, y)d\mu(z) - \int_{\Gamma^3} j_z(x, y)d\mu(z)d\mu(x)d\mu(y),$$

where j is the voltage function on Γ. Note that the second term is a triple integral, and therefore a constant dependent on Γ and μ.

Remark 5.2. As we saw in Remark 4.5, the voltage function is independent of choices of models and the corresponding parametrizations. Hence, Arakelov–Green functions are also independent of models and parametrizations.

Example 5.3. Integrals with respect to abstract measures can be quite complicated to work with. Let us consider an ostensibly easy example: let Γ be the complex unit circle from Example 2.13, and let

$$\mu_n := \sum_{k=0}^{n-1} a_k \delta_{e^{2k\pi i/n}}$$

for some $n \in \mathbb{N}$ and $a_k \in \mathbb{R}$ for all k, satisfying $\sum_{k=0}^{n-1} a_k = 1$. Then

$$\int_\Gamma j_z(x, y)d\mu_n(z) = \sum_{k=0}^{n-1} a_k \int_\Gamma j_z(x, y)d\delta_{e^{2k\pi i/n}}(z) = \sum_{k=0}^{n-1} a_k j_{e^{2k\pi i/n}}(x, y)$$

and

$$\int_{\Gamma^3} j_z(x, y)d\mu_n(z)d\mu_n(x)\mu_n(y) = \sum_{k,l,m=0}^{n-1} a_k a_l a_m j_{e^{2k\pi i/n}}\left(e^{2l\pi i/n}, e^{2m\pi i/n}\right),$$

so

$$g_{\mu_n}(x, y) = \sum_{k=0}^{n-1} a_k j_{e^{2k\pi i/n}}(x, y) - \sum_{k,l,m=0}^{n-1} a_k a_l a_m j_{e^{2k\pi i/n}}\left(e^{2l\pi i/n}, e^{2m\pi i/n}\right).$$

Expressions for the voltage function can then be substituted using Theorem 4.8 and Lemma 4.4.

Indeed, even with respect to a simple measure, computing the Arakelov–Green function can be awfully cumbersome. In the next sections, we consider measures that permit relatively nice expressions for the Arakelov–Green function, and describe methods to compute them.

Before we do so, however, we note a few important properties of general Arakelov–Green functions:

Proposition 5.4 (Properties of the Arakelov–Green function [2, Section 1.5])**.** *For any measure μ that satisfies the properties in Definition 5.1, the Arakelov–Green function g_μ is symmetric and piecewise smooth in both variables. Furthermore,*

$$\int_\Gamma g_\mu(x, y)d\mu(x) = 0 \quad \textit{and} \quad \Delta_x g_\mu(y) = \delta_y(x) - \mu(x).$$

5.2 The canonical measure

We now look at one specific measure, the **canonical measure** μ_{can}. This measure, discovered by Chinburg and Rumely in [4, Theorem 2.11], is not only important in various applications involving metrized graphs, but the corresponding Arakelov–Green function $g_{\mu_{can}}$ also has a relatively simple expression.

Theorem 5.5 (The canonical measure [2, Theorem 14.1 and Lemma 14.4])**.** *Let Γ be a metrized graph, with resistance function r. The* ***canonical measure***

$$\mu_{can} := \Delta_x \left(\tfrac{1}{2} r(x,y)\right) + \delta_y$$

is a non-negative measure independent of y, and it is the unique measure on Γ satisfying $\mu(\Gamma) = 1$ for which $g_\mu(x,x)$ is a constant. In fact,

$$g_{\mu_{can}}(x,y) = -\tfrac{1}{2} r(x,y) + \tau(\Gamma),$$

where $\tau(\Gamma)$ is the ***tau constant*** *of Γ, given by*

$$\tau(\Gamma) = \frac{1}{4} \int_\Gamma \left(\frac{\partial}{\partial x} r(x,y)\right)^2 dx,$$

so that $g_{\mu_{can}}(x,x) = \tau(\Gamma)$.

Remark 5.6. Since the Laplacian of a function is of the form in 2.21.1, the canonical measure is of the form in 2.21.1, so the canonical measure is a Borel measure.

Remark 5.7. The tau constant can be written more explicitly as

$$\begin{aligned}
\tau(\Gamma) &= \tfrac{1}{4} \int_\Gamma \left(\frac{\partial}{\partial x} r(x,y)\right)^2 dx \\
&= \tfrac{1}{4} \sum_{e \in E(\Gamma)} \int_e \left(\frac{\partial}{\partial x} r(x,y)\right)^2 dx|_e \\
&= \tfrac{1}{4} \sum_{e \in E(\Gamma)} \int_e \left(\frac{d}{dt} r(\varphi_e(t), y)\Big|_{\varphi_e(t)=x}\right)^2 dx|_e \\
&= \tfrac{1}{4} \sum_{e \in E(\Gamma)} \int_0^{\lambda_e} \left(\frac{d}{dt} r(\varphi_e(t), y)\right)^2 dt.
\end{aligned}$$

In Section 3.1 we saw that the quantity between parentheses is well-defined up to a sign, and thus its square is not dependent on our choice of orientations.

In [7], Cinkir gives a formula for the tau constant that only requires the discrete Laplacian matrix and its pseudo-inverse:

Theorem 5.8 (Formula for $\tau(\Gamma)$ [7, Theorem 1.1])**.** *Let Γ be a metrized graph with vertex set $V(\Gamma) = \{p_1, p_2, \ldots, p_n\}$. Denote the corresponding discrete Laplacian matrix by $L = (l_{ij})$ and its pseudo-inverse by $L^+ = (l^+_{ij})$. Then,*

$$\tau(\Gamma) = -\tfrac{1}{12} \sum_{\{p_i,p_j\} \in E(\Gamma)} l_{ij} \left(\frac{1}{l_{ij}} + l^+_{ii} - 2l^+_{ij} + l^+_{jj}\right)^2 + \tfrac{1}{4} \sum_{i,j=1}^{n} l_{ij} l^+_{ii} l^+_{jj} + \tfrac{1}{n} \operatorname{trace}\left(L^+\right).$$

Using the fact that $g_{\mu_{can}}(x,y) = -\frac{1}{2} r(x,y) + \tau(\Gamma)$ and the formulas from Theorem 4.8, we can express $g_{\mu_{can}}$ in terms of the resistance function's values at vertices.

Theorem 5.9 (Formulas for the Arakelov–Green function $g_{\mu_{can}}$ [8, Theorems 4.3 and 4.4])**.** *Let Γ be a metrized graph with vertex set $V(\Gamma)$ and edges with fixed orientations. Let $(p,q) \in E(\Gamma)$ and $x, y \in (p,q)$. If (p,q) is not a bridge, then*

$$g_{\mu_{can}}(x,y) = \tau(\Gamma) - \tfrac{1}{2}\,|x-y| + (x-y)^2\,\frac{\lambda_{(p,q)} - r(p,q)}{2\lambda_{(p,q)}^2}. \tag{5.9.1}$$

If (p,q) is a bridge, then

$$g_{\mu_{can}}(x,y) = \tau(\Gamma) - \tfrac{1}{2}\,|x-y|\,. \tag{5.9.2}$$

Now let $(p_1,q_1),(p_2,q_2) \in E(\Gamma)$ such that $\{p_1,q_1\} \neq \{p_2,q_2\}$, so that the edges are distinct but may share an end point, and let $x \in (p_1,q_1)$ and $y \in (p_2,q_2)$. If neither edge is a bridge, then

$$\begin{aligned} g_{\mu_{can}}(x,y) =& \tau(\Gamma) + x^2\frac{\lambda_{(p_1,q_1)} - r(p_1,q_1)}{2\lambda_{(p_1,q_1)}^2} + y^2\frac{\lambda_{(p_2,q_2)} - r(p_2,q_2)}{2\lambda_{(p_2,q_2)}^2} \\ &- \frac{xy}{\lambda_{(p_1,q_1)}\lambda_{(p_2,q_2)}}\left(j_{p_2}(p_1,q_2) - j_{p_2}(q_1,q_2)\right) - \frac{x}{2\lambda_{(p_1,q_1)}}\left(\lambda_{(p_1,q_1)} - 2j_{p_1}(q_1,p_2)\right) \\ &- \frac{y}{2\lambda_{(p_2,q_2)}}\left(\lambda_{(p_2,q_2)} - 2j_{p_2}(p_1,q_2)\right) - \tfrac{1}{2}r(p_1,p_2). \end{aligned} \tag{5.9.3}$$

If (p_1,q_1) is a bridge but (p_2,q_2) is not, then

$$g_{\mu_{can}}(x,y) = \begin{cases} \tau(\Gamma) + y^2\dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2)}{2\lambda_{(p_2,q_2)}^2} \\ \quad - y\dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2) + r(p_1,q_2) - r(p_1,p_2)}{2\lambda_{(p_2,q_2)}} & \text{if } (p_2,q_2) \subset \Gamma_{p_1}, \\ \quad - \tfrac{1}{2}\left(x + r(p_1,p_2)\right) \\ \\ \tau(\Gamma) + y^2\dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2)}{2\lambda_{(p_2,q_2)}^2} \\ \quad - y\dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2) + r(q_1,q_2) - r(q_1,p_2)}{2\lambda_{(p_2,q_2)}} & \text{if } (p_2,q_2) \subset \Gamma_{q_1}. \\ \quad - \tfrac{1}{2}\left(\lambda_{(p_1,q_1)} - x + r(q_1,p_2)\right) \end{cases} \tag{5.9.4}$$

Lastly, if both (p_1,q_1) and (p_2,q_2) are bridges, then

$$g_{\mu_{can}}(x,y) = \begin{cases} \tau(\Gamma) - \tfrac{1}{2}\left(x + y + r(p_1,p_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (p_1,p_2), \\ \\ \tau(\Gamma) - \tfrac{1}{2}\left(x + \lambda_{(p_2,q_2)} - y + r(p_1,q_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (p_1,q_2), \\ \\ \tau(\Gamma) - \tfrac{1}{2}\left(\lambda_{(p_1,q_1)} - x + y + r(q_1,p_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (q_1,p_2), \\ \\ \tau(\Gamma) - \tfrac{1}{2}\left(\lambda_{(p_1,q_1)} - x + \lambda_{(p_2,q_2)} - y + r(q_1,q_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (q_1,q_2). \end{cases} \tag{5.9.5}$$

From Theorem 5.9, we see that $g_{\mu_{can}}(x,y)$ can be described with a single formula when x and y are on fixed edges. Thus, for edges e_i and e_j of our metrized graph, there is a relatively simple expression for the function $z_{ij}\colon e_i \times e_j \to \mathbb{R}$ such that $z_{ij}(x,y) = g_{\mu_{can}}(x,y)$. This brings us to the following definition.

Definition 5.10 (Arakelov–Green matrix of $g_{\mu_{can}}$)**.** Let Γ be a metrized graph equipped with edge set

$$E(\Gamma) = \{(p_1,q_1),(p_2,q_2),\ldots,(p_n,q_n)\}.$$

The **Arakelov–Green matrix** of $g_{\mu_{can}}$ on Γ is given by the $n \times n$ matrix $Z := (z_{ij})$, where z_{ij} is a function such that $z_{ij}(x,y) = g_{\mu_{can}}(x,y)$ for $x \in (p_i,q_i)$ and $y \in (p_j,q_j)$.

Since the Arakelov–Green matrix has an entry z_{ij} for each pair of edges $(e_i, e_j) \in E(\Gamma)^2$, the matrix effectively encodes the entirety of $g_{\mu_{can}}$ on Γ. In fact, since each z_{ij} can be computed relatively easily by using Theorem 5.9, the Arakelov–Green matrix on Γ provides a method to efficiently compute $g_{\mu_{can}}$ on Γ. Note that this does require us to choose a vertex set for Γ, and to fix the orientations of the corresponding edges.

Since any Arakelov–Green function is symmetric, the Arakelov–Green matrix Z is symmetric too, in the sense that $Z(x, y) = Z^T(y, x)$.

5.3 Divisors and admissible metrics

The canonical measure discussed in the previous section is actually a specific case in a class of measures called **admissible metrics**. Before we discuss these measures though, we first look at **divisors** on metrized graphs.

Definition 5.11 (The divisor group on a metrized graph)**.** Let Γ be a metrized graph. The **divisor group** $\mathrm{Div}(\Gamma)$ is the set of formal sums

$$\sum_{p\in\Gamma} a_p p,$$

where each $a_p \in \mathbb{Z}$ and only finitely many a_p's are nonzero, equipped with the group operation

$$\sum_{p\in\Gamma} a_p p + \sum_{p\in\Gamma} b_p p := \sum_{p\in\Gamma} (a_p + b_p) p.$$

An element $D = \sum a_p p \in \mathrm{Div}(\Gamma)$ is called a **divisor** on Γ; its **support** is defined as

$$\mathrm{supp}(D) := \{p \in \Gamma \ : \ a_p \neq 0\}$$

and its **degree** is the integer

$$\deg(D) := \sum_{p\in\Gamma} a_p.$$

In this thesis, we do not look into the group structure of $\mathrm{Div}(\Gamma)$. Instead, we focus on individual divisors. An especially important divisor is the canonical divisor, whose application is discussed by Zhang in [17].

Definition 5.12 (Canonical divisor)**.** Let (Γ, ρ) be a polarized metrized graph as defined in Definition 2.25. The **canonical divisor** D_ρ of (Γ, ρ) is defined as

$$D_\rho := \sum_{p\in\Gamma} \left(v(p) - 2 + 2\rho(p)\right) p.$$

Example 5.13. Consider the metrized graph Γ in Figure 7, and define ρ to be 1 on p_2 and p_3, and 0 otherwise. Then the canonical divisor of (Γ, ρ) is

$$\begin{aligned} D_\rho &= \sum_{i\in\{2,3\}} \left(v(p_i) - 2 + 2\rho(p_i)\right) p_i \\ &= (3 - 2 + 2)p_2 + (1 - 2 + 2)p_3 \\ &= 3p_2 + p_3. \end{aligned}$$

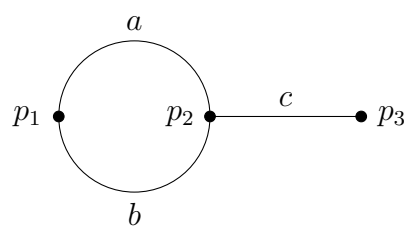


Figure 7: A metrized graph consisting of a circle and a line segment.

Definition 5.14 (Admissible metrics)**.** Let Γ be a metrized graph and let $D = \sum a_p p \in \operatorname{Div}(\Gamma)$ such that $\deg(D) \neq -2$. The **admissible metric** on Γ with respect to D is the measure given by

$$\mu_D := \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p\delta_p + 2\mu_{can}\right).$$

Proposition 5.15 (Properties of admissible metrics)**.** *Let Γ be a metrized graph and let $D = \sum a_p p \in \operatorname{Div}(\Gamma)$ such that $\deg(D) \neq -2$. Then $\mu_D(\Gamma) = 1$, $|\mu_D|(\Gamma) < \infty$, and if $D = 0$, then $\mu_D = \mu_{can}$.*

Proof. The first result follows from

$$\begin{aligned}\mu_D(\Gamma) &= \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p\delta_p(\Gamma) + 2\mu_{can}(\Gamma)\right)\\ &= \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p + 2\right)\\ &= \frac{1}{\deg(D)+2}\left(\deg(D)+2\right) = 1.\end{aligned}$$

As for the variation, let $S^+, S^- \in \mathcal{B}(\Gamma)$ be such that $S^+ \sqcup S^- = \Gamma$, and such that $\mu_D(S) \geq 0$ for all measurable $S \subset S^+$, and $\mu_D(S) \leq 0$ for all measurable $S \subset S^-$. Then

$$|\mu_D|(\Gamma) = \mu_D(S^+) - \mu_D(S^-).$$

Note that

$$\mu_D(S^+) = \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p\delta_p(S^+) + 2\mu_{can}(S^+)\right)$$

is finite, since both the sum and $\mu_{can}(S^+)$ are finite. Similarly, $\mu_D(S^-)$ is finite, so $|\mu_D|(\Gamma)$ is finite.

Lastly, let $D = 0$. That is, $D = \sum a_p p$ with $a_p = 0$ for all $p \in V(\Gamma)$. Then

$$\begin{aligned}\mu_D &= \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p\delta_p + 2\mu_{can}\right)\\ &= \frac{1}{0+2}\left(0 + 2\mu_{can}\right)\\ &= \mu_{can}.\end{aligned}$$

□

Thus, Proposition 5.15 tell us that for $D \in \operatorname{Div}(\Gamma)$, we have an Arakelov–Green function g_{μ_D}. Since μ_D is a generalization of μ_{can}, it may not come as a surprise that g_{μ_D}, too, has an explicit expression.

Theorem 5.16 (Expression for g_{μ_D} [5, Section 4.4])**.** *Let Γ be a metrized graph and let $D = \sum a_p p \in \operatorname{Div}(\Gamma)$ such that $\deg(D) \neq -2$. Then the Arakelov–Green function g_{μ_D} is given by*

$$g_{\mu_D}(x,y) = \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p j_p(x,y) + 4\tau(\Gamma) - r(x,y)\right) - c_{\mu_D}, \tag{5.16.1}$$

where c_{μ_D} is the constant

$$c_{\mu_D} = \frac{1}{2(\deg(D)+2)^2}\left(8\tau(\Gamma)(\deg(D)+1) + \sum_{p,q\in\Gamma} a_p a_q r(p,q)\right). \tag{5.16.2}$$

Remark 5.17. From Proposition 5.15, we know that $\mu_D = \mu_{can}$ for $D = 0$. Therefore, we have $g_{\mu_D} = g_{\mu_{can}}$ for $D = 0$. Let us check if the formula in Theorem 5.16 indeed coincides with the one from Theorem 5.5 in this special case. We start with the constant c_{μ_D}:

$$\begin{aligned} c_{\mu_D} &= \frac{1}{2(\deg(D)+2)^2}\left(8\tau(\Gamma)(\deg(D)+1) + \sum_{p,q\in\Gamma} a_p a_q r(p,q)\right) \\ &= \frac{1}{2(0+2)^2}\left(8\tau(\Gamma)(0+1)+0\right) \\ &= \frac{1}{8}\left(8\tau(\Gamma)\right) = \tau(\Gamma). \end{aligned}$$

Therefore,

$$\begin{aligned} g_{\mu_D}(x,y) &= \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p j_p(x,y) + 4\tau(\Gamma) - r(x,y)\right) - c_{\mu_D} \\ &= \frac{1}{0+2}\left(0 + 4\tau(\Gamma) - r(x,y)\right) - \tau(\Gamma) \\ &= -\tfrac{1}{2} r(x,y) + \tau(\Gamma) = g_{\mu_{can}}(x,y) \end{aligned}$$

as expected.

Note that, for any metrized graph Γ and a finite number of points on Γ, we can always choose a vertex set that contains all of those points. Hence, for any divisor $D \in \mathrm{Div}(\Gamma)$, we can equip Γ with a vertex set $V(\Gamma)$ such that $\mathrm{supp}(D) \subset V(\Gamma)$. This fact is vital in the next two theorems, which generalize the formulas from Theorem 5.9. Before we prove these results though, we first rewrite the expression from Theorem 5.16 into something more convenient.

Lemma 5.18 (Expression for g_{μ_D} without j)**.** *Let Γ be a metrized graph, $D = \sum a_p p \in \mathrm{Div}(\Gamma)$ such that $\deg(D) \neq -2$. Then*

$$g_{\mu_D}(x,y) = \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}\left(\kappa_D(x) + \kappa_D(y)\right)\right) - \tfrac{1}{2} r(x,y) - c_{\mu_D}, \tag{5.18.1}$$

where

$$\kappa_D(x) = \sum_{p\in\Gamma} a_p r(p,x) \tag{5.18.2}$$

and c_{μ_D} is as in 5.16.2.

Proof. Combining Theorem 5.16 with Lemma 4.4, we obtain

$$\begin{aligned} g_{\mu_D}(x,y) &= \frac{1}{\deg(D)+2}\left(\sum_{p\in\Gamma} a_p j_p(x,y) + 4\tau(\Gamma) - r(x,y)\right) - c_{\mu_D} \\ &= \frac{1}{\deg(D)+2}\left(\tfrac{1}{2}\sum_{p\in\Gamma} a_p\Big(r(p,x) + r(p,y) - r(x,y)\Big) + 4\tau(\Gamma) - r(x,y)\right) - c_{\mu_D} \\ &= \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}\sum_{p\in\Gamma} a_p r(p,x) + \tfrac{1}{2}\sum_{p\in\Gamma} a_p r(p,y) - \tfrac{1}{2}\sum_{p\in\Gamma} a_p r(x,y) - r(x,y)\right) - c_{\mu_D} \\ &= \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}\left(\kappa_D(x)+\kappa_D(y)\right) - \left(1 + \tfrac{1}{2}\sum_{p\in\Gamma} a_p\right) r(x,y)\right) - c_{\mu_D} \\ &= \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}\left(\kappa_D(x)+\kappa_D(y)\right) - \left(1 + \tfrac{1}{2}\deg(D)\right) r(x,y)\right) - c_{\mu_D} \\ &= \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}\left(\kappa_D(x)+\kappa_D(y)\right)\right) - \tfrac{1}{2} r(x,y) - c_{\mu_D}. \end{aligned}$$

□

Theorem 5.19 (Formulas for κ_D)**.** *Let Γ be a metrized graph and $D = \sum a_s s \in \operatorname{Div}(\Gamma)$. Equip Γ with a vertex set $V(\Gamma)$ such that $\operatorname{supp}(D) \subset V(\Gamma)$, and fix the orientations of the corresponding edges. Let $(p,q) \in E(\Gamma)$ and $x \in (p,q)$. If (p,q) is not a bridge, then*

$$\kappa_D(x) = \sum_{s \in V(\Gamma)} a_s \left(-x^2 \frac{\lambda_{(p,q)} - r(p,q)}{\lambda_{(p,q)}^2} + x \frac{\lambda_{(p,q)} - r(p,q) + r(s,q) - r(s,p)}{\lambda_{(p,q)}} + r(s,p) \right). \tag{5.19.1}$$

If (p,q) is a bridge, then

$$\kappa_D(x) = \sum_{s \in V(\Gamma) \cap \Gamma_p} a_s \left(x + r(s,p) \right) + \sum_{s \in V(\Gamma) \cap \Gamma_q} a_s \left(\lambda_{(p,q)} - x + r(s,q) \right). \tag{5.19.2}$$

Proof. This follows directly from applying Lemma 4.7 to 5.18.2. □

Theorem 5.20 (Formulas for the Arakelov–Green function g_{μ_D})**.** *Let Γ be a metrized graph and $D \in \operatorname{Div}(\Gamma)$ such that $\deg(D) \neq -2$. Equip Γ with a vertex set $V(\Gamma)$ such that $\operatorname{supp}(D) \subset V(\Gamma)$, fix the orientations of the corresponding edges, and put*

$$\chi_D(x,y) = \frac{1}{\deg(D)+2} \left(4\tau(\Gamma) + \tfrac{1}{2} \left(\kappa_D(x) + \kappa_D(y) \right) \right) - c_{\mu_D},$$

where for c_{μ_D} is as in 5.16.2 and κ_D as in 5.18.2. Let $(p,q) \in E(\Gamma)$ and $x, y \in (p,q)$. If (p,q) is not a bridge, then

$$g_{\mu_D}(x,y) = \chi_D(x,y) - \tfrac{1}{2} |x-y| + (x-y)^2 \frac{\lambda_{(p,q)} - r(p,q)}{2\lambda_{(p,q)}^2}. \tag{5.20.1}$$

If (p,q) is a bridge, then

$$g_{\mu_D}(x,y) = \chi_D(x) - \tfrac{1}{2} |x-y|. \tag{5.20.2}$$

Now let $(p_1,q_1), (p_2,q_2) \in E(\Gamma)$ such that $\{p_1,q_1\} \neq \{p_2,q_2\}$, so that the edges are distinct but may share an end point, and let $x \in (p_1,q_1)$ and $y \in (p_2,q_2)$. If neither edge is a bridge, then

$$\begin{aligned} g_{\mu_D}(x,y) = & \chi_D(x,y) + x^2 \frac{\lambda_{(p_1,q_1)} - r(p_1,q_1)}{2\lambda_{(p_1,q_1)}^2} + y^2 \frac{\lambda_{(p_2,q_2)} - r(p_2,q_2)}{2\lambda_{(p_2,q_2)}^2} \\ & - \frac{xy}{\lambda_{(p_1,q_1)}\lambda_{(p_2,q_2)}} \left(j_{p_2}(p_1,q_2) - j_{p_2}(q_1,q_2) \right) - \frac{x}{2\lambda_{(p_1,q_1)}} \left(\lambda_{(p_1,q_1)} - 2 j_{p_1}(q_1,p_2) \right) \\ & - \frac{y}{2\lambda_{(p_2,q_2)}} \left(\lambda_{(p_2,q_2)} - 2 j_{p_2}(p_1,q_2) \right) - \tfrac{1}{2} r(p_1,p_2). \end{aligned} \tag{5.20.3}$$

If (p_1,q_1) is a bridge but (p_2,q_2) is not, then

$$g_{\mu_D}(x,y) = \begin{cases} \chi_D(x,y) + y^2 \dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2)}{2\lambda_{(p_2,q_2)}^2} \\ \quad - y \dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2) + r(p_1,q_2) - r(p_1,p_2)}{2\lambda_{(p_2,q_2)}} & \text{if } (p_2,q_2) \subset \Gamma_{p_1}, \\ \quad - \tfrac{1}{2} \left(x + r(p_1,p_2) \right) \\ \\ \chi_D(x,y) + y^2 \dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2)}{2\lambda_{(p_2,q_2)}^2} \\ \quad - y \dfrac{\lambda_{(p_2,q_2)} - r(p_2,q_2) + r(q_1,q_2) - r(q_1,p_2)}{2\lambda_{(p_2,q_2)}} & \text{if } (p_2,q_2) \subset \Gamma_{q_1}. \\ \quad - \tfrac{1}{2} \left(\lambda_{(p_1,q_1)} - x + r(q_1,p_2) \right) \end{cases} \tag{5.20.4}$$

Lastly, if both (p_1, q_1) *and* (p_2, q_2) *are bridges, then*

$$g_{\mu_D}(x,y) = \begin{cases} \chi_D(x,y) - \frac{1}{2}\left(x + y + r(p_1,p_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (p_1,p_2), \\ \chi_D(x,y) - \frac{1}{2}\left(x + \lambda_{(p_2,q_2)} - y + r(p_1,q_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (p_1,q_2), \\ \chi_D(x,y) - \frac{1}{2}\left(\lambda_{(p_1,q_1)} - x + y + r(q_1,p_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (q_1,p_2), \\ \chi_D(x,y) - \frac{1}{2}\left(\lambda_{(p_1,q_1)} - x + \lambda_{(p_2,q_2)} - y + r(q_1,q_2)\right) & \text{if } \langle (p_1,q_1),(p_2,q_2)\rangle = (q_1,q_2). \end{cases} \tag{5.20.5}$$

Proof. This follows from substituting the formulas for $r(x,y)$ in Theorem 4.8 into 5.18.1. □

Remark 5.21. Notice that, if we compare the formulas in Theorem 5.20 with those in Theorem 5.9, we have for each case that

$$\begin{aligned} g_{\mu_D}(x,y) &= \chi_D(x,y) - \tau(\Gamma) + g_{\mu_{can}}(x,y) \\ &= \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}\left(\kappa_D(x) + \kappa_D(y)\right)\right) - \tau(\Gamma) + g_{\mu_{can}}(x,y) - c_{\mu_D}. \end{aligned}$$

If we put $D = 0$, then this reduces to

$$\begin{aligned} g_{\mu_D}(x,y) &= \frac{1}{0+2}\left(4\tau(\Gamma) + \tfrac{1}{2}(0+0)\right) - \tau(\Gamma) + g_{\mu_{can}}(x,y) - \tau(\Gamma) \\ &= g_{\mu_{can}}(x,y), \end{aligned}$$

as expected.

From Theorems 5.19 and 5.20, we see that, just like $g_{\mu_{can}}$, we can describe g_{μ_D} with one explicit formula for each pair of edges. Thus, we can generalize Definition 5.10:

Definition 5.22 (Arakelov–Green matrix of g_{μ_D})**.** Let Γ be a metrized graph and let $D \in \mathrm{Div}(\Gamma)$ such that $\deg(D) \neq -2$. Equip Γ with a vertex set $V(\Gamma)$ such that $\mathrm{supp}(D) \subset V(\Gamma)$, and enumerate

$$E(\Gamma) = \{(p_1,q_1),(p_2,q_2),\ldots,(p_n,q_n)\}.$$

The **Arakelov–Green matrix** of g_{μ_D} on Γ is given by the $n \times n$ matrix $Z_D := (z_{ij})$, where $z_{ij}(x,y) = g_{\mu_D}(x,y)$ for $x \in (p_i,q_i)$ and $y \in (p_j,q_j)$.

As with the Arakelov–Green matrix for $g_{\mu_{can}}$, we have $Z_D(x,y) = Z_D^T(y,x)$.

6 An algorithm to compute Arakelov–Green functions

6.1 Initialization

Let Γ be a metrized graph and equip it with a vertex set $V(\Gamma) = \{p_0, p_1, \ldots, p_{n-1}\}$ and enumerate the corresponding edge set $E(\Gamma) = \{e_0, e_1, \ldots, e_{m-1}\}^\dagger$. Each edge e_i has length λ_i and a fixed orientation (p_{e_i}, q_{e_i}).

With this construction, we can compute the discrete Laplacian matrix L using Definition 2.22, and its pseudo-inverse L^+ using Proposition 2.24

These matrices allow us to compute the resistance and voltage functions at vertices using Theorem 4.6, and the tau constant $\tau(\Gamma)$ using Theorem 5.8.

6.2 The connectivity matrix

An essential part of our algorithm is the connectivity matrix, which encodes all information about the structure of Γ that we need in order to apply Theorems 5.19 and 5.20.

Let $e_i, e_j \in E(\Gamma)$ and suppose that e_i is a bridge. Then the function α encodes whether $e_j \subset \Gamma_{p_{e_i}}$ or $e_j \subset \Gamma_{q_{e_i}}$ with a single bit:

$$\alpha(e_i, e_j) := \begin{cases} 0 & \text{if } e_j \subset \Gamma_{p_{e_i}}, \\ 1 & \text{if } e_j \subset \Gamma_{q_{e_i}}; \end{cases}$$

If e_j is also a bridge, then β encodes the closest neighbours of e_i and e_j with two bits:

$$\beta(e_i, e_j) := \begin{cases} 0 & \text{if } \langle e_i, e_j \rangle = (p_{e_i}, p_{e_j}), \\ 1 & \text{if } \langle e_i, e_j \rangle = (p_{e_i}, q_{e_j}), \\ 10 & \text{if } \langle e_i, e_j \rangle = (q_{e_i}, p_{e_j}), \\ 11 & \text{if } \langle e_i, e_j \rangle = (q_{e_i}, q_{e_j}). \end{cases}$$

Now, the connectivity matrix $C = (c_{ij})$ is an $m \times m$ matrix with binary entries defined as follows:

$$c_{ij} := \begin{cases} 0 & \text{if neither } e_i \text{ nor } e_j \text{ is a bridge;} \\ 1 & \text{if } i = j \text{ and } e_i \text{ is a bridge;} \\ \alpha(e_i, e_j) & \text{if } e_i \text{ is a bridge but } e_j \text{ is not;} \\ \alpha(e_j, e_i) & \text{if } e_j \text{ is a bridge but } e_i \text{ is not;} \\ 100\alpha(e_i, e_j) + \beta(e_i, e_j) & \text{if } i \neq j, \text{ and both } e_i \text{ and } e_j \text{ are bridges.} \end{cases}$$

Given two edges $e_i, e_j \in E(\Gamma)$, the entries c_{ii} and c_{jj} tell us which of them are bridges. If one of the edges (say, e_i) is a bridge and the other (e_j) is not, the single bit in c_{ij} encodes whether e_j is part of the subgraph $\Gamma_{p_{e_i}}$ or $\Gamma_{q_{e_i}}$.

If both edges are bridges (so if $c_{ii} = c_{jj} = 1$), c_{ij} consists of three bits: the first bit tells us if $e_j \subset \Gamma_{p_{e_i}}$ or $e_j \subset \Gamma_{q_{e_i}}$, while the second and third bits encode the closest neighbours of the two edges.

Example 6.1. Consider the metrized graph with the model depicted in Figure 8; notice that this is the same metrized graph we saw in Example 2.20. As we have six edges, our connectivity matrix C will be a 6×6 matrix.

The edges e_0 and e_5 are bridges, so $c_{00} = c_{55} = 1$, while the other diagonal entries are 0. Now, we need to determine $\alpha(0, i)$ for each $i \in \{1, 2, \ldots, 5\}$ and $\alpha(5, i)$ for each $i \in \{0, 1, \ldots, 4\}$.

We have $e_1 \subset \Gamma_{p_1}$, and since $q_{e_0} = p_1$, we have $\alpha(0, 1) = 1$, so $c_{01} = c_{10} = 1$. Similarly, $e_1 \subset \Gamma_{p_4}$, so $\alpha(5, 1) = 0$, and thus $c_{51} = c_{15} = 0$. We can exploit this symmetry for any pair of edges with one bridge.

†In this and the next section we start our indices at 0, as is typical for programming languages.

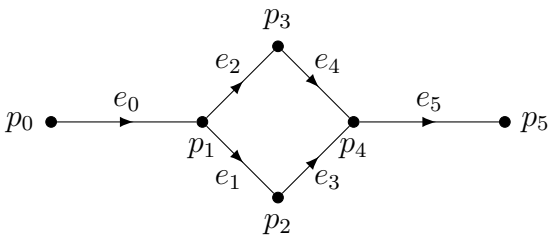


Figure 8: A model for a metrized graph with two bridges.

The entries c_{05} and c_{50} are more complicated. We have that $e_5 \subset \Gamma_{p_1}$, so $\alpha(0,5) = 1$. The closest neighbours of the bridges are given by $\langle e_0, e_5 \rangle = (p_1, p_4)$, so $\beta(0,5) = 10$. This yields

$$c_{05} = 100\alpha(0,5) + \beta(0,5) = 110.$$

Computing all c_{ij}'s gives us the connectivity matrix

$$C = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 110 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

We can now quickly look up aspects of the metrized graph's structure: since $c_{00} = 1$ and $c_{33} = 0$, we know that e_0 is a bridge but e_3 is not. Since $c_{03} = 1$, we can also tell that $e_3 \subset \Gamma_{q_{e_0}}$.

The pseudocode in Algorithm 6.1 describes how to generate the connectivity matrix.

Algorithm 6.1: Generating the connectivity matrix

Input: Metrized graph Γ with $V(\Gamma) = \{p_0, p_1, \ldots, p_{n-1}\}$ and $E(\Gamma) = \{e_0, e_1, \ldots, e_{m-1}\}$
Output: Connectivity matrix $C = (c_{ij})$

```
C := 0_{m×m};                    /* Initialize C as an m × m matrix with zero entries */
for i ∈ {0, 1, ..., m − 1} do                                  /* Set diagonal entries */
    if e_i is a bridge then
        c_ii ← 1;

for i ∈ {0, 1, ..., m − 1} do                              /* Set non-diagonal entries */
    if c_ii = 1 then                                                  /* If e_i is a bridge */
        for j ∈ {i + 1, i + 2, ..., m − 1} do                /* Loop over the next edges e_j */
            c_ij ← α(e_i, e_j);                             /* Assume that e_j is not a bridge */
            if c_jj = 1 then                  /* If e_j is a bridge, adjust the entries */
                c_ij ← 100c_ij + β(e_i, e_j);
                c_ji ← 100α(e_j, e_i) + β(e_j, e_i);
            else                             /* If e_j is not a bridge, set c_ji = α(e_i, e_j) */
                c_ji ← c_ij
    else                                                              /* If e_i is not a bridge */
        for j ∈ {i + 1, i + 2, ..., m − 1} do                /* Loop over the next edges e_j */
            if c_jj = 1 then          /* If e_j is a bridge, put the appropriate entries */
                c_ij ← α(e_j, e_i);
                c_ji ← c_ij;
```

6.3 Computing κ_D on each edge

Since our vertices are enumerated, we can express any divisor $D \in \mathrm{Div}(\Gamma)$ with $\mathrm{supp}(D) \subset V(\Gamma)$ as $D = \sum_{k=0}^{n-1} a_k p_k$. In fact, we can treat such divisors as vectors of n entries:

$$D = (a_0, a_1, \ldots, a_{n-1}).$$

This allows us to compute the constant c_{μ_D} with the formula

$$c_{\mu_D} = \frac{1}{2(\deg(D)+2)^2}\left(8\tau(\Gamma)(\deg(D)+1) + \sum_{i,j=0}^{n-1} a_i a_j r(p_i, p_j)\right).$$

Using Theorem 5.19, Algorithm 6.2 determines a formula for κ_D on a given edge e_i: a function $K_i \colon e_i \to \mathbb{R}$ such that $K_i(x) = \kappa_D(x)$ for $x \in e_i$. Repeating this algorithm for each edge yields an m-vector $K = (K_i)$, which completely describes κ_D on Γ.

Algorithm 6.2: Computing κ_D

Input: Metrized graph Γ with $V(\Gamma) = \{p_0, p_1, \ldots, p_{n-1}\}$ and $E(\Gamma) = \{e_0, e_1, \ldots, e_{m-1}\}$
Integer $i \in \{0, 1, \ldots, m-1\}$
Connectivity matrix $C = (c_{ij})$
Divisor $D = (a_0, a_1, \ldots, a_{n-1})$ on Γ

Output: Function K_i, which satisfies $K_i(x) = \kappa_D(x)$ for $x \in e_i$

```
K_i := 0;                                                    /* Initialize K_i */
if c_ii = 0 then                                             /* If e_i is not a bridge, use 5.19.1 */
    for k ∈ {0, 1, ..., n − 1} do
        K_i(x) ← K_i(x) + a_k r(p_k, p_{e_i})
                 + a_k ( − x^2 (λ_i − r(p_{e_i}, q_{e_i})) / λ_i^2 + x (λ_i − r(p_{e_i}, q_{e_i}) + r(p_k, q_{e_i}) − r(p_k, p_{e_i})) / λ_i );
else                                                         /* If e_i is a bridge, use 5.19.2 */
    for k ∈ {0, 1, ..., n − 1} do                            /* Iterate over all vertices */
        if p_k = p_{e_i} then
            K_i(x) ← K_i(x) + a_k x;
        else if p_k = q_{e_i} then
            K_i(x) ← K_i(x) + a_k (λ_i − x);
        else                                                 /* If p_k is not an end point of e_i */
            for j ∈ {0, 1, ..., m − 1} \ {i} do              /* Iterate over all other edges */
                if p_k is an end point of e_j then           /* Until we find an edge with end point
                 p_k */
                    if (c_jj = 0 and c_ij = 0) or (c_jj = 1 and c_ij < 100) then
                        K_i(x) ← K_i(x) + a_k (x + r(p_k, p_{e_i}));      /* Case p_k ∈ Γ_{p_{e_i}} */
                    else
                        K_i(x) ← K_i(x) + a_k (λ_i − x + r(p_k, q_{e_i}));  /* Case p_k ∈ Γ_{q_{e_i}} */
                    break;
```

6.4 Computing the Arakelov–Green matrix

With κ_D encoded in K, we can now compute the Arakelov–Green matrix $Z = (z_{ij})$ defined in 5.22, using the formulas in Theorem 5.20. The process to compute z_{ij} is described in Algorithm 6.3; repeating this algorithm for each i and j yields the matrix Z. We use the shorthand

$$\chi_{ij}^D(x, y) := \frac{1}{\deg(D)+2}\left(4\tau(\Gamma) + \tfrac{1}{2}(K_i(x) + K_j(y))\right) - c_{\mu_D}.$$

In Section A.1, the full algorithm to compute Z is implemented as the method `ag_matrix()` of the class `MetrizedGraph`.

Algorithm 6.3: Computing the Arakelov–Green matrix

Input: Metrized graph Γ with $V(\Gamma) = \{p_0, p_1, \ldots, p_{n-1}\}$ and $E(\Gamma) = \{e_0, e_1, \ldots, e_{m-1}\}$
Integers $i, j \in \{0, 1, \ldots, m-1\}$
Connectivity matrix $C = (c_{ij})$
Divisor D on Γ
Vector $K = (K_i)$ as computed in Algorithm 6.2

Output: Function z_{ij}, which satisfies $z_{ij}(x, y) = g_{\mu_D}(x, y)$ for $x \in e_i$ and $y \in e_j$

if $i = j$ **then** `/* Same edge */`

  if $c_{ii} = 0$ **then** `/* Same edge, not a bridge: 5.20.1 */`

    $$z_{ii}(x,y) \leftarrow \chi_{ij}^D(x,y) - \tfrac{1}{2}|x-y| + (x-y)^2 \frac{\lambda_i - r(p_{e_i}, q_{e_i})}{2\lambda_i^2};$$

  else `/* Same edge, bridge: 5.20.2 */`

    $$z_{ii}(x,y) \leftarrow \chi_{ij}^D(x,y) - \tfrac{1}{2}|x-y|;$$

else `/* Different edges */`

  if $c_{ii} = 0$ *and* $c_{jj} = 0$ **then** `/* If neither is a bridge: 5.20.3 */`

    $$\begin{aligned} z_{ij}(x,y) \leftarrow & \chi_{ij}^D(x,y) + x^2 \frac{\lambda_i - r(p_{e_i}, q_{e_i})}{2\lambda_i^2} + y^2 \frac{\lambda_j - r(p_{e_j}, q_{e_j})}{2\lambda_j^2} \\ & - \frac{xy}{\lambda_i \lambda_j}\left(j_{p_{e_j}}(p_{e_i}, q_{e_j}) - j_{p_{e_j}}(q_{e_i}, q_{e_j})\right) - \frac{x}{2\lambda_i}\left(\lambda_i - 2j_{p_{e_i}}(q_{e_i}, p_{e_j})\right) \\ & - \frac{y}{2\lambda_j}\left(\lambda_j - 2j_{p_{e_j}}(p_{e_i}, q_{e_j})\right) - \tfrac{1}{2} r(p_{e_i}, p_{e_j}); \end{aligned}$$

  else if $c_{ii} = 1$ *and* $c_{jj} = 0$ **then** `/* If` e_i `is and` e_j `is not a bridge: 5.20.4 */`

    if $c_{ij} = 0$ **then** `/* If` $e_j \subset \Gamma_{p_{e_i}}$ `*/`

    $$\begin{aligned} z_{ij}(x,y) \leftarrow & \chi_{ij}^D(x,y) + y^2 \frac{\lambda_j - r(p_{e_j}, q_{e_j})}{2\lambda_j^2} - y \frac{\lambda_j - r(p_{e_j}, q_{e_j}) + r(p_{e_i}, q_{e_j}) - r(p_{e_i}, p_{e_j})}{2\lambda_j} \\ & - \tfrac{1}{2}\left(x + r(p_{e_i}, p_{e_j})\right); \end{aligned}$$

    else `/* If` $e_j \subset \Gamma_{q_{e_i}}$ `*/`

    $$\begin{aligned} z_{ij}(x,y) \leftarrow & \chi_{ij}^D(x,y) + y^2 \frac{\lambda_j - r(p_{e_j}, q_{e_j})}{2\lambda_j^2} - y \frac{\lambda_j - r(p_{e_j}, q_{e_j}) + r(q_{e_i}, q_{e_j}) - r(q_{e_i}, p_{e_j})}{2\lambda_j} \\ & - \tfrac{1}{2}\left(\lambda_{e_i} - x + r(q_{e_i}, p_{e_j})\right); \end{aligned}$$

  else if $c_{ii} = 0$ *and* $c_{jj} = 1$ **then** `/* If` e_i `is not and` e_j `is an bridge: 5.20.4 */`

    if $c_{ij} = 0$ **then** `/* If` $e_j \subset \Gamma_{p_{e_i}}$ `*/`

    $$\begin{aligned} z_{ij}(x,y) \leftarrow & \chi_{ij}^D(x,y) + x^2 \frac{\lambda_i - r(p_{e_i}, q_{e_i})}{2\lambda_i^2} - x \frac{\lambda_i - r(p_{e_i}, q_{e_i}) + r(p_{e_j}, q_{e_i}) - r(p_{e_j}, p_{e_i})}{2\lambda_i} \\ & - \tfrac{1}{2}\left(y + r(p_{e_j}, p_{e_i})\right); \end{aligned}$$

    else `/* If` $e_j \subset \Gamma_{q_{e_i}}$ `*/`

    $$\begin{aligned} z_{ij}(x,y) \leftarrow & \chi_{ij}^D(x,y) + x^2 \frac{\lambda_i - r(p_{e_i}, q_{e_i})}{2\lambda_i^2} - x \frac{\lambda_i - r(p_{e_i}, q_{e_i}) + r(q_{e_j}, q_{e_i}) - r(q_{e_j}, p_{e_i})}{2\lambda_i} \\ & - \tfrac{1}{2}\left(\lambda_{e_j} - y + r(q_{e_j}, p_{e_i})\right); \end{aligned}$$

  else `/* If both edges are bridges: 5.20.5 */`

    case $c_{ij} \bmod 100 \equiv 0$ **do** `/* If` $\langle e_i, e_j \rangle = (p_{e_i}, p_{e_j})$ `*/`

    $$z_{ij}(x,y) \leftarrow \chi_{ij}^D(x,y) - \tfrac{1}{2}\left(x + y + r(p_{e_i}, p_{e_j})\right);$$

    case $c_{ij} \bmod 100 \equiv 1$ **do** `/* If` $\langle e_i, e_j \rangle = (p_{e_i}, q_{e_j})$ `*/`

    $$z_{ij}(x,y) \leftarrow \chi_{ij}^D(x,y) - \tfrac{1}{2}\left(x + \lambda_j - y + r(p_{e_i}, q_{e_j})\right);$$

    case $c_{ij} \bmod 100 \equiv 10$ **do** `/* If` $\langle e_i, e_j \rangle = (q_{e_i}, p_{e_j})$ `*/`

    $$z_{ij}(x,y) \leftarrow \chi_{ij}^D(x,y) - \tfrac{1}{2}\left(\lambda_i - x + y + r(q_{e_i}, p_{e_j})\right);$$

    case $c_{ij} \bmod 100 \equiv 11$ **do** `/* If` $\langle e_i, e_j \rangle = (q_{e_i}, q_{e_j})$ `*/`

    $$z_{ij}(x,y) \leftarrow \chi_{ij}^D(x,y) - \tfrac{1}{2}\left(\lambda_i - x + \lambda_j - y + r(q_{e_i}, q_{e_j})\right);$$

6.5 Consistency checks

Although there is no easy way the verify the correctness of our algorithm without scrutinizing the proof of Theorem 5.20 and the formulas used in Algorithms 6.1, 6.2 and 6.3, we can do two quick tests to check if particular results from this algorithm coincide with other information that we discussed previously. Neither gives us complete certainty that our algorithm is correct, but they do allow us to identify errors.

Firstly, a function is well-defined if its value is the same on the same point, regardless of how that point is represented. Using the notation introduced at the end of Section 2.1, a vertex p with $v(p) > 1$ can be represented in at least two ways: if, for instance, we have edges $e_1 = (p, q)$ and $e_2 = (s, p)$, both $(e_1, 0)$ and (e_2, λ_2) represent the same point p. Let p' be another vertex, represented as $(e_3, 0)$. Then, since Arakelov–Green functions are well-defined functions, we must have

$$z_{13}(0, 0) = g_{\mu_D}(p, p') = z_{23}(\lambda_2, 0).$$

This test can be generalized to check any pair of vertices where at least one vertex has a valence above 1. An algorithm for this test is implemented in Section A.2.

As for another consistency check, recall the formula from Theorem 5.16. Using Theorem 4.6, we can immediately compute the function values of g_{μ_D} on vertices; of course, these values ought to match the values computed with the Arakelov–Green matrix generated by our algorithm. A script that runs this test on all vertices of a given metrized graph is provided in Section A.3.

7 Computational examples

In this section, we discuss several computational examples and use the Sage code from Section A.1 to compute the Arakelov–Green matrix. In these examples, I denotes the identity matrix and J denotes the matrix or vector whose entries are all 1; the sizes of these matrices can be determined from the context.

7.1 The circle graph

Consider once again the metrized graph from Example 2.13. We wish to compute the Arakelov–Green matrix of $g_{\mu_{can}}$, which from Proposition 5.15 we know is g_{μ_D} for $D = 0$. Hence, we can use the model from Example 2.2.

Enumerate the vertices as $(1, i, -1)$ and the edges as

$$(e_0, e_1, e_2) = \Big((1, i), (i, -1), (-1, 1)\Big).$$

Then $\lambda_0 = \lambda_1 = \frac{1}{2}\pi$ and $\lambda_2 = \pi$. The corresponding discrete Laplacian matrix and its pseudo-inverse are

$$L = \frac{1}{\pi}\begin{bmatrix} 3 & -2 & -1 \\ -2 & 4 & -2 \\ -1 & -2 & 3 \end{bmatrix} \quad \text{and} \quad L^+ = \frac{\pi}{72}\begin{bmatrix} 11 & -4 & -7 \\ -4 & 8 & -4 \\ -7 & -4 & 11 \end{bmatrix}.$$

The tau constant is $\tau(\Gamma) = \frac{1}{6}\pi$, and the connectivity matrix is the 3×3 zero matrix, since this metrized graph has no bridges. The Arakelov–Green matrix, then, is given by

$$Z(x, y) = \frac{\pi}{48}\begin{bmatrix} 8 & -1 & -4 \\ -1 & 8 & -1 \\ -4 & -1 & 8 \end{bmatrix} - \frac{1}{4}\begin{bmatrix} 2|x-y| & -x+y & 0 \\ x-y & 2|x-y| & -x+y \\ 0 & x-y & 2|x-y| \end{bmatrix} + 12(x-y)^2 J.$$

This matches [8, Section 6, Example I].

7.2 Joint circles

Let C_1 and C_2 denote two circles, with arc length ℓ_1 and ℓ_2 respectively, which touch one another at the point p_0; see Figure 9. Define the metrized graph Γ to be the circles' union, and let a divisor on Γ be given by $D = 2p_0$.

Of course, a suitable vertex set for Γ must contain more than just p_0, lest the model has self-loops and multiple edges. The model that we use is depicted in Figure 9, with edges enumerated and oriented as shown. As for the lengths of the edges, we have $\lambda_0 = \lambda_1 = \lambda_2 = \ell_1/3$ and $\lambda_3 = \lambda_4 = \lambda_5 = \ell_2/3$.

The discrete Laplacian matrix for this vertex set is

$$L = \frac{3}{\ell_1\ell_2}\begin{bmatrix} 2\ell_1 + 2\ell_2 & -\ell_2 & -\ell_2 & -\ell_1 & -\ell_1 \\ -\ell_2 & 2\ell_2 & -\ell_2 & 0 & 0 \\ -\ell_2 & -\ell_2 & 2\ell_2 & 0 & 0 \\ -\ell_1 & 0 & 0 & 2\ell_1 & -\ell_1 \\ -\ell_1 & 0 & 0 & -\ell_1 & 2\ell_1 \end{bmatrix},$$

and its pseudo-inverse is

$$L^+ = \frac{\ell_1}{225}\begin{bmatrix} 6 & -9 & -9 & 6 & 6 \\ -9 & 26 & 1 & -9 & -9 \\ -9 & 1 & 26 & -9 & -9 \\ 6 & -9 & -9 & 6 & 6 \\ 6 & -9 & -9 & 6 & 6 \end{bmatrix} + \frac{\ell_2}{225}\begin{bmatrix} 6 & 6 & 6 & -9 & -9 \\ 6 & 6 & 6 & -9 & -9 \\ 6 & 6 & 6 & -9 & -9 \\ -9 & -9 & -9 & 26 & 1 \\ -9 & -9 & -9 & 1 & 26 \end{bmatrix}.$$

The tau constant of this metrized graph is $\tau(\Gamma) = \frac{1}{12}(\ell_1 + \ell_3)$. The formulas for κ_D, for $D = (2, 0, 0, 0, 0)$ are listed in K:

Figure 9: A metrized graph consisting of two circles joint at p_0 (left), and one of its models (right).

$$K(x) = \frac{1}{9\ell_1\ell_2}\begin{bmatrix} 18\ell_1\ell_2 x - 18\ell_2 x^2 \\ 4\ell_1^2\ell_2 + 6\ell_1\ell_2 x - 2\ell_2 x^2 \\ 4\ell_1^2\ell_2 - 6\ell_1\ell_2 x - 2\ell_2 x^2 \\ 18\ell_1\ell_2 x - 18\ell_1 x^2 \\ 4\ell_1\ell_2^2 + 6\ell_1\ell_2 x - 2\ell_1 x^2 \\ 4\ell_1\ell_2^2 - 6\ell_1\ell_2 x - 2\ell_1 x^2 \end{bmatrix}.$$

The Arakelov–Green matrix of g_{μ_D}, then is

$$Z_D(x,y) = \begin{bmatrix} M_1(x,y) & W(x,y) \\ W^T(y,x) & M_2(x,y) \end{bmatrix}.$$

where

$$\begin{aligned} M_i(x,y) =& \frac{\ell_j}{48}J + \frac{\ell_i}{144}\begin{bmatrix} 3 & -5 & -5 \\ -5 & 19 & 3 \\ -5 & 3 & 19 \end{bmatrix} + \frac{1}{12}\begin{bmatrix} 3x+3y & 5x-y & x+y \\ -x+5y & x+y & 3x-3y \\ x+y & -3x+3y & -x-y \end{bmatrix} \\ & - \frac{1}{2}|x-y|I + \frac{1}{4\ell_i}\left(x^2+y^2-4xy\right)J \end{aligned}$$

with $j \in \{1,2\} \setminus \{i\}$, and

$$\begin{aligned} W(x,y) =& \frac{\ell_1}{144}\begin{bmatrix} 3 & 3 & 3 \\ -5 & -5 & -5 \\ -5 & -5 & -5 \end{bmatrix} + \frac{\ell_2}{144}\begin{bmatrix} 3 & -5 & -5 \\ 3 & -5 & -5 \\ 3 & -5 & -5 \end{bmatrix} - \frac{1}{12}\begin{bmatrix} 3x+3y & 3x+y & 3x-y \\ x+3y & x+y & x-y \\ -x+3y & -x+y & -x-y \end{bmatrix} \\ & + \frac{1}{4\ell_1\ell_2}\left(\ell_2 x^2 + \ell_1 y^2\right)J. \end{aligned}$$

The 36 entries of Z_D coincide with the formulas that Moriwaki found in [13, Section 3].

7.3 A canonical divisor

Recall the polarized metrized graph from Example 5.13, pictured in Figure 7. We found that the corresponding canonical divisor is $D_\rho = 3p_2 + p_3$. By strategically choosing one more vertex p_0, we obtain the suitable model in Figure 10. The lengths of the edges are given by $\lambda_0 = \lambda_1 = a/2$, $\lambda_2 = b$ and $\lambda_3 = c$.

The discrete Laplacian matrix is

$$L = \frac{1}{abc}\begin{bmatrix} 4bc & -2bc & -2bc & 0 \\ -2bc & (a+2b)c & -ac & 0 \\ -2bc & -ac & ab+ac+2bc & -ac \\ 0 & 0 & -ab & ab \end{bmatrix}$$

with pseudo-inverse

$$L^+ = \frac{a}{64(a+b)}\begin{bmatrix} 9a+10b & -3a+2b & -3a-6b & -3a-6b \\ -3a+2b & a+26b & a-14b & a-14b \\ -3a-6b & a-14b & a+10b & a+10b \\ -3a-6b & a-14b & a+10b & a+10b \end{bmatrix} + \frac{c}{16}\begin{bmatrix} 1 & 1 & 1 & -3 \\ 1 & 1 & 1 & -3 \\ 1 & 1 & 1 & -3 \\ -3 & -3 & -3 & 9 \end{bmatrix}.$$

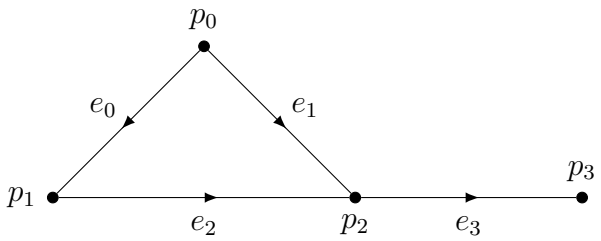


Figure 10: A model for the metrized graph in Figure 7.

And thus our tau constant is $\tau(\Gamma) = \frac{1}{12}(a+b+3c)$. For our divisor $D_\rho = (0,0,3,1)$, we have the following Arakelov–Green matrix:

$$Z_{D_\rho}(x,y) = \frac{c}{9}J + \frac{1}{216(a+b)}\begin{bmatrix} M(x,y) & V(x,y) \\ V^T(y,x) & W(x,y) \end{bmatrix},$$

where M is the 3×3 matrix

$$\begin{aligned} M(x,y) = &\begin{bmatrix} 38a^2+76ab+2b^2 & 38a^2+76ab+2b^2 & -7a^2+58ab+2b^2 \\ 38a^2+76ab+2b^2 & 38a^2+76ab+2b^2 & -7a^2+58ab+2b^2 \\ -7a^2+58ab+2b^2 & -7a^2+58ab+2b^2 & 2a^2+148ab+2b^2 \end{bmatrix} \\ &+ 36\begin{bmatrix} 2bx+2by & -(3a+b)x-(3a+5b)y & 5by-(2a+b)y \\ -(3a+5b)x-(3a+b)y & -2bx-2by & -5bx-(2a+b)y \\ -(2a+b)x+5by & (2a+b)x-5by & -2(a-b)x-2(a-b)y \end{bmatrix} \\ &- 108(a+b)|x-y|I + 36(x^2+y^2)J + 216xy\begin{bmatrix} -1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & -1 \end{bmatrix}, \end{aligned}$$

V is the 3-vector

$$V(x,y) = \begin{bmatrix} -7a^2-14ab+2b^2 \\ -7a^2-14ab+2b^2 \\ 2a^2-32ab+2b^2 \end{bmatrix} + 36x\begin{bmatrix} -b \\ b \\ a-b \end{bmatrix} + 36\left(x^2-2(a+b)y\right)J,$$

and

$$W(x,y) = 2a^2+4ab+2b^2+36(a+b)\left(x+y-3|x-y|\right).$$

Now, let $x = (e_1, a/9)$ and $y = (e_3, c/2)$. Then the function value at (x,y) is encoded in the $(1,3)$-th entry of Z_{D_ρ}:

$$\begin{aligned} g_{\mu_{D_\rho}}(x,y) &= \frac{c}{9} + \frac{1}{216(a+b)}V_1(x,y) \\ &= \frac{c}{9} + \frac{1}{216(a+b)}\Big(-7a^2-14ab+2b^2+36bx+36\left(x^2-2(a+b)y\right)\Big) \\ &= \frac{c}{9} + \frac{1}{216(a+b)}\Big(-7a^2-14ab+2b^2+36b\cdot a/9+36\left((a/9)^2-2(a+b)\cdot c/2\right)\Big) \\ &= \frac{c}{9} + \frac{1}{216(a+b)}\Big(-7a^2-14ab+2b^2+4ab+36\left(\tfrac{1}{81}a^2-(a+b)c\right)\Big) \\ &= \frac{c}{9} + \frac{1}{216(a+b)}\left(-\tfrac{59}{9}a^2-10ab+2b^2-36(a+b)c\right) \\ &= -\frac{1}{1944}\left(59a^2+90ab-18b^2\right) - \frac{c}{18}. \end{aligned}$$

7.4 The epsilon invariant on a tesseract

We now discuss a more involved application of the Arakelov–Green matrix: computing invariants ϵ_D of a metrized graph. This invariant is thoroughly discussed in [5, Section 4], and for our purposes is defined as follows:

Definition 7.1 (Epsilon invariants)**.** Let Γ be a metrized graph and let $D = \sum a_p p \in \mathrm{Div}(\Gamma)$ such that $\deg(D) \neq -2$. Then

$$\epsilon_D := (\deg(D) + 2) \sum_{p \in \Gamma} a_p g_{\mu_D}(p, x) + \sum_{p \in \Gamma} a_p r(p, x), \tag{7.1.1}$$

which is independent of $x \in \Gamma$.

A method to compute ϵ_D is given in Algorithm 7.1, which exploits the fact that 7.1.1 is independent of our choice of x. In this case, we choose $x = p_{e_0}$.

Algorithm 7.1: Computing the epsilon invariant

Input: Metrized graph Γ with $V(\Gamma) = \{p_0, p_1, \ldots, p_{n-1}\}$ and $E(\Gamma) = \{e_0, e_1, \ldots, e_{m-1}\}$
Divisor $D = (a_0, a_1, \ldots, a_{n-1})$ on Γ
Arakelov–Green matrix $Z = (z_{ij})$

Output: Invariant ϵ_D

```
ε_D := 0;                                                    /* Initialize ε_D */
/* Compute the term of 7.1.1 involving g_{μ_D}                                 */
for i ∈ {0, 1, ..., n − 1} do                                /* For each vertex, */
    for k ∈ {0, 1, ..., m − 1} do        /* Find an edge of which it is an end point */
        if p_i = p_{e_k} then            /* If p_i = p_{e_k}, then g_{μ_D}(p_{e_0}, p_i) = z_{0k}(0, 0) */
            g ← z_{0k}(0, 0);
            break;
        else if p_i = q_{e_k} then       /* If p_i = q_{e_k}, then g_{μ_D}(p_{e_0}, p_i) = z_{0k}(0, λ_k) */
            g ← z_{0k}(0, λ_k);
            break
        ε_D ← ε_D + a_i g;
ε_D ← (deg(D) + 2)ε_D;
/* Compute and add the other term                                              */
ε_D ← ε_D + Σ_{i=0}^{n−1} a_i r(p_i, p_{e_0});
```

In [5, Theorem 4.27], Cinkir found an alternative expression, which does not involve Arakelov–Green functions and is easier to compute:

$$\epsilon_D = \frac{1}{\deg(D) + 2} \left(4\tau(\Gamma) \deg(D) + \sum_{p,q \in \Gamma} a_p a_q r(p, q) \right). \tag{7.1.2}$$

A Sage implementation of Algorithm 7.1, as well as of Cinkir's formula, can be found in Section A.1 as the `epsilon()` method.

We now compute ϵ_D on the metrized graph formed by a tesseract using both 7.1.1 and 7.1.2, and check if the formulas indeed coincide. The vertices and edges of Γ are the usual vertices and edges of a tesseract, see Figure 11. More concretely,

$$V(\Gamma) = \big\{ (b_0, b_1, b_2, b_3) \ : \ b_i \in \{0, 1\} \big\} \subset \mathbb{R}^4,$$

where we enumerate the sixteen vertices as

$$(b_0, b_1, b_2, b_3) = p_{b_0 + 2b_1 + 4b_2 + 8b_3}.$$

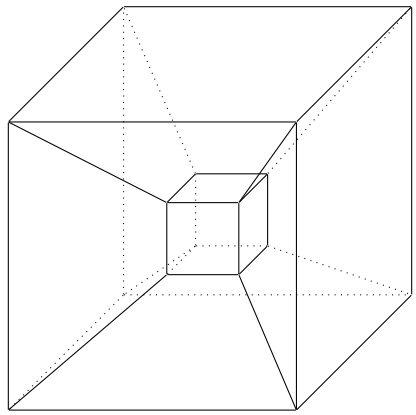


Figure 11: A three-dimensional representation of a tesseract.

The corresponding edge set is then given by

$$E(\Gamma) = \big\{(p_i, p_j) \ : \ i < j \text{ and } |p_i - p_j| = 1\big\},$$

and each edge has length 1. For the divisor

$$D = \sum_{i=0}^{15} i p_i,$$

we compute ϵ_D directly through Algorithm 7.1 and by using Cinkir's formula. Unsurprisingly, the two methods yield the same result: $\epsilon_D = 7875/122$.

7.5 The epsilon invariant on a banana graph

Consider the metrized graph Γ pictured alongside one of its models in Figure 12. Lengths of edges are given by $\lambda_0 = b$, $\lambda_1 = \lambda_2 = a/2$ and $\lambda_3 = \lambda_4 = c/2$. We equip Γ with a polarization ρ defined as $\rho(p) = 0$ for all $p \in \Gamma$, so the canonical divisor of (Γ, ρ) is

$$D_\rho = p_0 + p_1.$$

The invariant ϵ_{D_ρ} was computed by Moriwaki in [12, Section 3, Type VII], but Cinkir found in [5, Example 4.35 and Remark 4.36] that Moriwaki's formula is wrong. The expression that Cinkir found instead is

$$\epsilon_{D_\rho} = \tfrac{1}{6}\left(a + b + c + \frac{abc}{ab + bc + ac}\right).$$

When we compute ϵ_{D_ρ} via Algorithm 7.1 or Cinkir's formula 7.1.2 (i.e. via `epsilon()`), we obtain the same result.

If $a = b = c$, then

$$\epsilon_{D_\rho} = \frac{5}{9}a,$$

which matches Moriwaki's original formula.

Figure 12: A metrized "banana" graph (left), and one of its models (right).

8 Conclusion

We began this thesis by defining metrized graphs, which we equipped with familiar structures such as a metric and a σ-algebra. After studying piecewise functions in general, we focused on voltage and resistance functions. Although these functions were defined in a rather contrived manner, Theorem 4.8 gave us a relatively straightforward method to compute their values.

We then looked at Arakelov–Green functions, quickly learned how cumbersome it is to compute them for general measures, and resorted to the special case of the canonical measure. We summarized Cinkir's formulas for this case in Theorem 5.9 and then sought to generalize these expressions for any admissible metric. Although the resulting formulas in Theorem 5.20 are longer, they, like Cinkir's original theorem, only require information about the metrized graph's structure that can easily be obtained.

Theorem 5.20 allowed us to construct the algorithm described in Section 6, which we implemented in Sage in Section A.1 and applied to numerical and symbolic examples in Section 7.

Although applications of Arakelov–Green functions are beyond the scope of this thesis, the algorithm we devised to compute Arakelov–Green matrices can be valuable for further research. For instance, the `MetrizedGraph` class from Section A.1 and its `ag_matrix()` method can be used in a more complex program to derive polarized metrized graphs from algebraic curves, and then compute the Arakelov–Green matrix for the canonical divisor.

We computed the epsilon invariant in Sections 7.4 and 7.5, but there are other important invariants on metrized graphs that are defined using Arakelov–Green functions, such as the phi and lambda invariants [9]. Our algorithm can assist in computing these invariants, too.

Furthermore, since metrized graphs have applications beyond abstract mathematics, `MetrizedGraph` can be expanded upon to include methods relevant to their uses in physics and biology.

References

[1] Baker, M., and Faber, X. Metrized graphs, Laplacian operators, and electrical networks. *Contemporary Mathematics 415*, 2 (2006), 15–33.

[2] Baker, M., and Rumely, R. Harmonic analysis on metrized graphs. *Canadian Journal of Mathematics 59*, 2 (2007), 225–275.

[3] Bapat, R. *Graphs and Matrices*, 2nd ed. Springer, 2014.

[4] Chinburg, T., and Rumely, R. The capacity pairing. *Journal für die reine und angewandte Mathematik 434* (1993), 1–44.

[5] Cinkir, Z. *The tau constant of metrized graphs.* PhD thesis, University of Georgia, 2007.

[6] Cinkir, Z. Generalized Foster's identities. *International Journal of Quantum Chemistry 111*, 10 (2011), 2228–2233.

[7] Cinkir, Z. The tau constant and the discrete Laplacian matrix of a metrized graph. *European Journal of Combinatorics 32*, 4 (2011), 639–655.

[8] Cinkir, Z. Explicit computation of certain Arakelov–Green functions. *Kyoto Journal of Mathematics 54*, 4 (2013), 759–774.

[9] Cinkir, Z. Admissible invariants of genus 3 curves. *Manuscripta Mathematica 148* (2015), 317–339.

[10] Cohn, D. *Measure Theory.* Birkhäuser, 1980.

[11] Harary, F. *Graph Theory.* Taylor & Francis Group, 1969.

[12] Moriwaki, A. Bogomolov conjecture for curves of genus 2 over function fields. *Kyoto Journal of Mathematics 36*, 4 (1996), 687–695.

[13] Moriwaki, A. Bogomolov conjecture over function fields for stable curves with only irreducible fibers. *Compositio Mathematica 105*, 2 (1997), 125–140.

[14] Rao, C., and Mitra, S. *Generalized Inverse of Matrices and its Applications.* John Wiley & Sons, 1971.

[15] Sutherland, W. *Introduction to Metric & Topological Spaces.* Oxford University Press, 2009.

[16] Vos, V. S. S. Methods for determining the effective resistance. Master's thesis, Universiteit Leiden, 2016.

[17] Zhang, S. Admissible pairings on a curve. *Inventiones Mathematicae 112*, 1 (1993), 171–193.

A Sage code

A.1 The MetrizedGraph class

```
# file   :   arakelov_green.sage
# author:   Ruben van Dijk
# date   :   Tue 25 May 2021

# Description:
# A collection of mathematical tools related to metrized graphs. Most
# importantly, this  includes the MetrizedGraph class and a method to
# compute its Arakelov-Green function g_muD, given a divisor D.

var('x','y')

# METRIZED GRAPH CLASS
class MetrizedGraph(DiGraph):
    """Metrized Graph.

A metrized graph is an equivalence class of parametrized graphs, see
"Computing Arakelov-Green functions on Metrized Graphs". This implementation
equips a metrized graph with a fixed (user-specified) vertex set, edge set
and orientations.

INPUT:

A metrized graph is initialized with MetrizedGraph(V,E), where:

    V   is an array of enumerated vertices;

    E   is an array of enumerated edges, which are encoded in the form
        (p,q,l). Here, p is the first point of the edge, q the second,
        and l the length.

ATTRIBUTES:

Upon initialization, several attributes are computed:

    L       The discrete Laplacian matrix of the metrized graph

    Lplus   The Moore-Penrose pseudo-inverse of L

    C       The connectivity matrix of the metrized graph

METHODS:

The following methods are available:

    vertex(i)       Return the ith vertex

    edge(i)         Return the ith edge

    p(i)            Return the first vertex of the ith edge

    q(i)            Return the second vertex of the ith edge

    length(i)       Return the length of the ith edge

    total_length()  Compute the total length of the metrized graph

    valence(p)      Compute the valence of a vertex p

    resistance(p,q) Compute the resistance between vertices p and q

    voltage(p,q,s)  Compute the voltage function j_s(p,q) on vertices

    tau()           Compute the tau constant of the metrized graph

    can_divisor()   Compute the canonical divisor of the metrized graph

    kappa(D)        Compute the kappa_D function on each edge

    ag_matrix(D)    Compute the Arakelov-Green matrix of g_muD

    epsilon(D)      Compute the epsilon invariant for the divisor D"""

    def __init__(self, V, E):
        # Store the vertices, edges, and the order they were given in.
        self._order_vertices = V
```

```
        self._order_edges = E

        # Inherit everything from the DiGraph class
        super().__init__([V, E], weighted = True)

        # Compute discrete Laplacian matrix and pseudo-inverse
        self.L = discrete_laplacian_matrix(self)
        self.Lplus = moore_penrose(self.L).simplify_rational()

        # Compute connectivity matrix
        self.C = connectivity(self)

    def __repr__(self):
        v = self.order()
        e = self.size()
        return "Metrized graph with %s vertices and %s edges" %(v, e)

    # Basic attributes that use the order in which vertices and edges
    # were given
    def vertex(self, i):
        return self._order_vertices[i]

    def edge(self, i):
        return self._order_edges[i]

    def p(self,i):
        return self._order_edges[i][0]

    def q(self,i):
        return self._order_edges[i][1]

    def length(self, i):
        return self._order_edges[i][2]

    def total_length(self):
        return sum(length(i) for i in range(self.size()))

    # Valence of a vertex
    def valence(self,p):
        val = 0

        for n in range(self.size()):
            if p in (self.p(n), self.q(n)):
                val += 1

        return val

    # Resistance function on vertices
    def resistance(self,p,q):
        i = self._order_vertices.index(p)
        j = self._order_vertices.index(q)
        resist = self.Lplus[i][i] - 2*self.Lplus[i][j] + self.Lplus[j][j]
        return resist.full_simplify()

    # Voltage function on vertices
    def voltage(self,p,q,s):
        i = self._order_vertices.index(p)
        j = self._order_vertices.index(q)
        k = self._order_vertices.index(s)
        volt = self.Lplus[k][k]
        volt += -self.Lplus[i][k] - self.Lplus[j][k] + self.Lplus[i][j]
        return volt.full_simplify()

    # Tau constant
    def tau(self):
        L = self.L
        Lplus = self.Lplus

        tau = -1/12 * sum(L[i][j]*(1/L[i][j] + Lplus[i][i] - 2*Lplus[i][j] +
            Lplus[j][j])^2 for i in range(self.order()) for j in range(self.
            order()) if self.has_edge((self.vertex(i),self.vertex(j))))
        tau += 1/4 * sum(L[i][j]*Lplus[i][i]*Lplus[j][j] for i in range(self.
            order()) for j in range(self.order()))
        tau += 1/self.order() * Lplus.trace()

        return tau.full_simplify()
```

```
    # Canonical divisor
    def can_divisor(self,rho=None):
        """Computes the canonical divisor of the metrized graph given
a polarization on each vertex.

The canonical divisor of a metrized graph is the divisor whose coefficient
corresponding to vertex p is valence(p) - 2 + 2*polarization(p). The
polarization is encoded as an array where the ith entry corresponds to
the value on the ith vertex. If no polarization is specified, all values
are assumed to be 0."""

        if rho == None:
            rho = [0 for i in range(self.order())]
        else:
            assert len(rho) == self.order()

        D = [self.valence(self.vertex(i)) - 2 + 2*rho[i] for i in range(self.
            order())]

        return D

    # Kappa-array per edge; uses parallelization
    def kappa(self, D = None):
        """Computes formulas for kappa_D on each edge.

If the divisor D is not specified, D = 0 is used."""

        if D is None:
            D = [0 for i in range(self.order())]
        else:
            assert len(D) == self.order()

        K = [0 for i in range(self.size())]
        generator = kappa([(self, i, D) for i in range(self.size())])
        for k in generator:
            K[k[0][0][1]] = k[1]

        return K

    # Arakelov-Green matrix for g_muD; uses parallelization
    def ag_matrix(self,D=None):
        """Computes the Arakelov-Green matrix for g_muD.

The (i,j)th entry of the Arakelov-Green matrix is a formula for g_muD(x,y)
when x lies on the ith and y lies on the jth edge. If the divisor D is not
specified, D = 0 is used, yielding the Arakelov-Green matrix for the
Arakelov-Green function with respect to the canonical measure."""

        if D is None:
            D = [0 for i in range(self.order())]
        else:
            assert len(D) == self.order()

        # Determine Kappa-array
        K = self.kappa(D)

        # Compute tau constant and deg(D)
        tau = self.tau()
        deg = sum(D)

        # Compute constant cD
        cD = 8*tau*(deg+1)
        cD += sum(D[i]*D[j]*self.resistance(self.vertex(i), self.vertex(j))
            for i in range(self.order()) for j in range(self.order()))
        cD *= 1 / (2 * (deg+2)^2)

        # Compute Arakelov-Green matrix
        Z = matrix(SR, self.size())
        generator = Arakelov_Green([(self,i,j,D,deg,K,tau,cD) for i in range(
            self.size()) for j in range(i,self.size())])
        for g in generator:
            (i,j) = (g[0][0][1], g[0][0][2])
            Z[i,j] = g[1]

            if i != j:
                Z[j,i] = Z[i,j].substitute(x=y,y=x)

        return Z
```

```
    # Epsilon invariant
    def epsilon(self, D=None, method='thm'):
        """Computes the epsilon invariant for a given divisor.

This function can use two different methods: def, which uses the definition
of the epsilon invariant using Arakelov-Green functions; or thm, which uses
the theorem by Cinkir. If no method is specified, thm is used."""

        assert method == 'thm' or method == 'def'
        assert len(D) == self.order()

        if method == 'thm':
            eps = sum(D[i]*D[j]*G.resistance(G.vertex(i),G.vertex(j)) for i in range(G.order()) for j in range(G.order()))
            eps += 4 * G.tau() * sum(D)
            eps /= sum(D) + 2
            return eps.full_simplify()

        else:
            Z = G.ag_matrix(D)
            eps = 0

            for i in range(G.order()):
                v = G.vertex(i)

                for k in range(G.size()):
                    if v is G.p(k):
                        g = Z[0,k](x=0,y=0)
                        break
                    elif v is G.q(k):
                        g = Z[0,k](x=0,y=G.length(k))
                        break

                eps += D[i] * g

            eps *= sum(D) + 2
            eps += sum(D[i]*G.resistance(G.vertex(i),G.p(0)) for i in range(G.order()))

            return eps.full_simplify()


# DISCRETE LAPLACIAN MATRIX
# Function to compute the discrete Laplacian matrix of a metrized graph G,
# using the definition of the discrete Laplacian matrix.
def discrete_laplacian_matrix(G):
        L = matrix(SR,G.order())

        for i in range(G.order()):
            for j in range(i+1,G.order()):
                if Graph(G).has_edge((G.vertex(i),G.vertex(j))):
                    L[i,j] = -1/Graph(G).edge_label(G.vertex(i),G.vertex(j))
                    L[j,i] = L[i,j]

            L[i,i] = -sum(L[i,k] for k in range(G.order()) if k != i)

        return L

# Function to compute the Moore-Penrose pseudo-inverse of a discrete
# Laplacian matrix.
def moore_penrose(M):
    n = M.nrows()
    return (M - (1/n)*matrix.ones(n)).inverse() + (1/n)*matrix.ones(n)



# CONNECTIVITY MATRIX
# Helper function alpha. Given a bridge e_i and an edge e_j of metrized graph
# G, its output is 0 if e_j is in G_{p_{e_i}} and 1 if e_j is in G_{q_{e_i}}.
def alpha(G,i,j):
    # Delete e_i to obtain G - e_i
    G.delete_edge(G._order_edges[i])

    # Determine which connected subgraph p_{e_j} belongs to; this is the
    # subgraph that e_j belongs to.
    if G.p(i) in G.connected_component_containing_vertex(G.p(j)):
        alpha = 0
    else:
        alpha = 1
```

```
    # Restore e_i and return result
    G.add_edge(G._order_edges[i])
    return alpha

# Helper function beta. Given two bridges e_i and e_j of metrized graph G,
# it outputs the  bridges' closest neighbours: 0 if the first points of
# e_i and e_j are closest  neighbours, 10 if the second point of e_i and
# the first point of e_j are closest neighbours, etcetera.
def beta(G,i,j):
    # Temporarily set the lengths of e_i and e_j to 1, and the lengths of
    # other edges to 0. The closest neighbours of e_i and e_j are then
    # points on the respective edges that are distance 0 apart from
    # each other.
    for k in range(G.size()):
        if k == i or k == j:
            G.set_edge_label(G.p(k), G.q(k), 1)
        else:
            G.set_edge_label(G.p(k), G.q(k), 0)

    # Determine the closest neighbours
    for index in range(2):
        for jndex in range(2):
            dist = Graph(G).shortest_path_length(G.edge(i)[index], G.edge(j)[jndex], by_weight = True)
            if dist == 0:
                # Restore the edge lengths and return result
                for k in range(G.size()):
                    G.set_edge_label(G.p(k), G.q(k), G.length(k))
                return 10*index + jndex

# Function to compute the connectivity matrix of metrized graph G
def connectivity(G):
    # Initialize C
    C = matrix(G.size())

    # Set diagonal entries
    for i in range(G.size()):
        if G.is_cut_edge(G.edge(i)):
            C[i,i] = 1

    # Set non-diagonal entries
    for i in range(G.size()):
        if C[i,i] == 1:                     # If e_i is a bridge,
            for j in range(i+1,G.size()):   # Loop over the next edges e_j
                C[i,j] = alpha(G,i,j)       # Assume that e_j is not a bridge
                if C[j,j] == 1:             # If e_j is a bridge, adjust
                    C[i,j] = 100*C[i,j] + beta(G,i,j)
                    C[j,i] = 100*alpha(G,j,i) + beta(G,j,i)
                else:
                    C[j,i] = C[i,j]         # If e_j is not a bridge,
                                            # set c_ji=c_ij

        else:                               # If e_i is not a bridge,
            for j in range(i+1,G.size()):   # Loop over the next edges e_j
                if C[j,j] == 1:             # If e_j is a bridge, set
                    C[i,j] = alpha(G,j,i)   # nonzero entry
                    C[j,i] = C[i,j]

    # Return result
    return C


# KAPPA
# Given a divisor D, this function determines a formula for kappa on the
# edge e_i of the metrized graph G. With the @parallel decorator, this
# function can make use of multiple processors when it is applied to
# multiple edges.
@parallel
def kappa(G, i, D):
    # Initialize kappa
    kappa = 0

    # If e_i is not a bridge, use eq 5.19.1
    if G.C[i,i] == 0:
        for k in range(G.order()):
                tmp = -x^2 * (G.length(i) - G.resistance(G.p(i), G.q(i))) / (G.length(i)^2)
                tmp += x * (G.length(i) - G.resistance(G.p(i), G.q(i)) + G.
```

```
                    resistance(G.vertex(k), G.q(i)) - G.resistance(G.vertex(k), G.p(i))) / G.length(i)
                tmp += G.resistance(G.vertex(k), G.p(i))
                tmp *= D[k]
                kappa += tmp

    # If e_i is a bridge, use eq 5.19.2
    else:
        # Iterate over all vertices, add the right expression to kappa
        # for each vertex
        for k in range(G.order()):
            if G.vertex(k) == G.p(i):
                kappa += D[k]*x
            elif G.vertex(k) == G.q(i):
                kappa += D[k]*(G.length(i) - x)
            else:
                # If vertex k is not an end point of e_i, search for an edge
                # that the vertex is an end point of. Then use the
                # connectivity matrix to determine the suitable expression.
                for j in range(G.size()):
                    if j != i and (G.vertex(k) == G.p(j) or G.vertex(k) == G.q(j)):
                        if (G.C[j,j] == 0 and G.C[i,j] == 0) or (G.C[j,j] == 1 and G.C[i,j] < 100):
                            kappa += D[k] * (x + G.resistance(G.vertex(k), G.p(i)))
                        else:
                            kappa += D[k] * (G.length(i) - x + G.resistance(G.vertex(k), G.q(i)))
                        break

    # Return result
    return kappa


# FORMULAS FOR A-G FUNCTION
# Given a divisor D, this function determines a formula for the
# Arakelov-Green function g_muD on the pair of edges (e_i,e_j)
# of the metrized graph G. The input also requires several constants
# that should be computed beforehand. With the @parallel decorator,
# this function can make use of multiple processors when it is applied
# to multiple pairs of edges.
@parallel
def Arakelov_Green(G, i, j, D, deg, K, tau, cD):
    # Short-hand for clunky recurring expression
    chi = lambda i,j: 1/(deg+2) * (4*tau + 1/2 * (K[i] + K[j].substitute(x=y))) - cD

    # Same edge
    if i == j:
        if G.C[i,i] == 0:   # Not a bridge: eq 5.20.1
            g = (G.length(i) - G.resistance(G.p(i), G.q(i))) / (2 * G.length(i)^2)
            g = chi(i,i) - 1/2*abs(x-y) + (x-y)^2 * g
        else:               # Bridge: eq 5.20.2
            g = chi(i,i) - 1/2*abs(x-y)

    # Different edges
    else:
        # Neither a bridge: eq 5.20.3
        if G.C[i,i] == 0 and G.C[j,j] == 0:
                g = chi(i,j)
                g += x^2 * (G.length(i) - G.resistance(G.p(i),G.q(i))) / (2 * G.length(i)^2)
                g += y^2 * (G.length(j) - G.resistance(G.p(j),G.q(j))) / (2 * G.length(j)^2)
                g += -(x*y)/(G.length(i)*G.length(j)) * (G.voltage(G.p(i), G.q(j), G.p(j)) - G.voltage(G.q(i), G.q(j), G.p(j)))
                g += -x/(2*G.length(i)) * (G.length(i) - 2*G.voltage(G.q(i), G.p(j), G.p(i)))
                g += -y/(2*G.length(j)) * (G.length(j) - 2*G.voltage(G.p(i), G.q(j), G.p(j)))
                g += -1/2*G.resistance(G.p(i), G.p(j))

        # Edge e_i a bridge, e_j not: eq 5.20.4
        elif G.C[i,i] == 1 and G.C[j,j] == 0:
            if G.C[i,j] == 0:                   # If e_j in G_{p_{e_i}}
                g = chi(i,j) + y^2 * (G.length(j) - G.resistance(G.p(j),G.q(j))) / (2*G.length(j)^2)
```

```
                g += -y * (G.length(j) - G.resistance(G.p(j),G.q(j)) + G.resistance(G.p(i),G.q(j)) - G.resistance(G.p(i),G.p(j))) / (2*G.length(j))
                g += -1/2*(x + G.resistance(G.p(i),G.p(j)))

            else:                                   # If e_j in G_{q_{e_i}}
                g = chi(i,j) + y^2 * (G.length(j) - G.resistance(G.p(j),G.q(j))) / (2*G.length(j)^2)
                g += -y * (G.length(j) - G.resistance(G.p(j),G.q(j)) + G.resistance(G.q(i),G.q(j)) - G.resistance(G.q(i),G.p(j))) / (2*G.length(j))
                g += -1/2*(G.length(i) - x + G.resistance(G.q(i),G.p(j)))

        # Edge e_j a bridge, e_i not: adapted eq 5.20.4
        elif G.C[i,i] == 0 and G.C[j,j] == 1:
            if G.C[i,j] == 0:                       # If e_i in G_{p_{e_j}}
                g = chi(i,j) + x^2 * (G.length(i) - G.resistance(G.p(i),G.q(i)))/(2*G.length(i)^2)
                g += -x * (G.length(i) - G.resistance(G.p(i),G.q(i)) + G.resistance(G.p(j),G.q(i)) - G.resistance(G.p(j),G.p(i))) / (2*G.length(i))
                g += -1/2*(y + G.resistance(G.p(j),G.p(i)))

            else:                                   # If e_i in G_{q_{e_j}}
                g = chi(i,j) + x^2 * (G.length(i) - G.resistance(G.p(i),G.q(i)))/(2*G.length(i)^2)
                g += -x * (G.length(i) - G.resistance(G.p(i),G.q(i)) + G.resistance(G.q(j),G.q(i)) - G.resistance(G.q(j),G.p(i))) / (2*G.length(i))
                g += -1/2*(G.length(j) - y + G.resistance(G.q(j),G.p(i)))

        # If both edges are bridges: eq 5.20.5, using the closest-neighbour
        # information encoded in the connectivity matrix
        else:
            if G.C[i,j] % 100 == 0:
                g = chi(i,j) - 1/2*(x + y + G.resistance(G.p(i),G.p(j)))

            elif G.C[i,j] % 100 == 1:
                g = chi(i,j) - 1/2*(x + G.length(j) - y + G.resistance(G.p(i),G.q(j)))

            elif G.C[i,j] % 100 == 10:
                g = chi(i,j) - 1/2*(G.length(i) - x + y + G.resistance(G.q(i),G.p(j)))

            else:
                g = chi(i,j) - 1/2*(G.length(i) - x + G.length(j) -  y + G.resistance(G.q(i),G.q(j)))

    # Simplify and return result
    return g.full_simplify()
```

A.2 Well-definedness test

```
# file   :   edge_point_test.sage
# author:   Ruben van Dijk
# date   :   Thu 10 June 2021

# Description:
# Simple script that computes the Arakelov-Green matrix of a metrized graph
# and checks if the function values on vertices are the same regardless of
# the vertices' representations, verifying that the Arakelov-Green function
# is well-defined.

load('arakelov_green.sage')

# A big weird metrized graph with bridges, non-bridges, and different
# directions, for illustration.
var('v1','v2','v3','v4','v5','v6','v7','v8','v9','v10','w1','w2','w3','w4','w5','w6','w7','w8')

V = [v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, w1, w2, w3, w4, w5, w6, w7, w8]
E = [(v1,v2,12), (v1,v6,312), (v2,v7,123), (v4,v3,1321), (v4,v5,7777), (v4,v6,1), (v7,v6,11), (v7,v5,1321321), (v7,v8,2), (v8,v2,19), (v2,v3,8), (v8,v3,9), (v8,v4,1233), (v1,v9,2123), (v9,v10,321), (v10,w1,432), (w1,w2,1), (w2,w3,1), (w2,w4,2), (w3,w4,1), (w4,w1,1), (w2,w5,2), (w5,w6,10), (w6,w7,1), (w6,w8,4), (w7,w8,1)]

G = MetrizedGraph(V,E)

# Just some weird divisor, for the sake of illustration.
D = [(i^3+14) % 27 for i in range(len(V))]

Z = G.ag_matrix(D)

test = []

for i in range(G.order()):
    for j in range(G.order()):
        v = G.vertex(i)
        w = G.vertex(j)

        F = []

        for nv in range(G.size()):
            for nw in range(G.size()):
                f = Z[nv,nw]

                if v is G.p(nv) and w is G.p(nw):
                    F.append( f(x=0, y=0) )

                elif v is G.p(nv) and w is G.q(nw):
                    F.append( f(x=0, y=G.length(nw)) )

                elif v is G.q(nv) and w is G.p(nw):
                    F.append( f(x=G.length(nv), y=0) )

                elif v is G.q(nv) and w is G.q(nw):
                    F.append( f(x=G.length(nv), y=G.length(nw)) )

        if F == [F[0] for i in range(len(F))]:
            test.append(0)
        else:
            print(F)
            print()
            test.append(1)

if sum(test) == 0:
    print('Success!')
else:
    print(test)
```

A.3 Vertex value test

```
# file   :   vertex_value_test.sage
# author:   Ruben van Dijk
# date   :   Thu 10 June 2021

# Description:
# Script that computes the Arakelov-Green matrix of a metrized graph and
# checks if the function values on vertices match the function values
# obtained through direct computation.

load('arakelov_green.sage')

# The Arakelov-Green function on vertices
def gmuD(G,D,x,y):
    tau = G.tau()

    g = sum(D[i]*G.voltage(x,y,G.vertex(i)) for i in range(G.order()))
    g += 4*tau - G.resistance(x,y)
    g *= 1/(sum(D) + 2)

    cD = 8*tau*(sum(D)+1)
    cD += sum(D[i]*D[j]*G.resistance(G.vertex(i), G.vertex(j)) for i in range
        (G.order()) for j in range(G.order()))
    cD *= 1 / (2 * (sum(D)+2)^2)

    return g - cD

# A big weird metrized graph with bridges, non-bridges, and different
# directions, for illustration.
var('v1','v2','v3','v4','v5','v6','v7','v8','v9','v10','w1','w2','w3','w4','
    w5','w6','w7','w8')

V = [v1, v2, v3, v4, v5, v6, v7, v8, v9, v10, w1, w2, w3, w4, w5, w6, w7, w8]
E = [(v1,v2,12), (v1,v6,312), (v2,v7,123), (v4,v3,1321), (v4,v5,7777), (v4,v6
    ,1), (v7,v6,11), (v7,v5,1321321), (v7,v8,2), (v8,v2,19), (v2,v3,8), (v8,v3
    ,9), (v8,v4,1233), (v1,v9,2123), (v9,v10,321), (v10,w1,432), (w1,w2,1), (
    w2,w3,1), (w2,w4,2), (w3,w4,1), (w4,w1,1), (w2,w5,2), (w5,w6,10), (w6,w7
    ,1), (w6,w8,4), (w7,w8,1)]

G = MetrizedGraph(V,E)

# Just some weird divisor, for the sake of illustration.
D = [(i^3+14) % 27 for i in range(len(V))]

Z = G.ag_matrix(D)

test = []

for i in range(G.order()):
    for j in range(G.order()):
        v = G.vertex(i)
        w = G.vertex(j)

        true_value = gmuD(G,D,v,w)

        F = []

        for nv in range(G.size()):
            for nw in range(G.size()):
                f = Z[nv,nw]

                if v is G.p(nv) and w is G.p(nw):
                    F.append( f(x=0, y=0) )

                elif v is G.p(nv) and w is G.q(nw):
                    F.append( f(x=0, y=G.length(nw)) )

                elif v is G.q(nv) and w is G.p(nw):
                    F.append( f(x=G.length(nv), y=0) )

                elif v is G.q(nv) and w is G.q(nw):
                    F.append( f(x=G.length(nv), y=G.length(nw)) )

        if F == [true_value for i in range(len(F))]:
            test.append(0)
        else:
            print(F)
            test.append(1)
```

```python

if sum(test) == 0:
    print('Success!')
else:
    print(test)
```