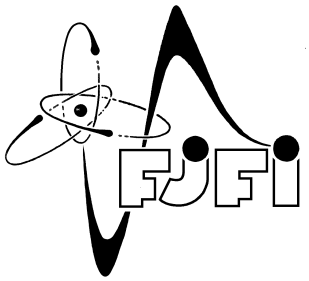
Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering



DIPLOMA WORK

Spectra Of Quantum Graphs

Gabriela Malenová
Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering

Supervision:

Pavel Kurasov
Department of Mathematics, Stockholm University,
Stockholm, Sweden

David Krejčiřík
Nuclear Physics Institute in Řež, Academy of Sciences,
Prague, Czech Republic

Acknowledgments

Special thanks to:

Pavel Kurasov (Stockholm University), for excellent help, advice and mentoring of the project and countless suggestions along the way. His personal support was absolutely invaluable, which I am very grateful for.

David Krejčiřík (Academy of Sciences of the Czech Republic), for mentoring the thesis, inspiring suggestions, kind support and for carefully reading the drafts.

Erik Wernersson (Uppsala University), for valuable help while struggling with the numerics in the early stages.

Nick Hale (Oxford University), for great deal of amazing work on Chebfun and kind support while implementing the code.

Prohlášení

Prohlašuji, že jsem svou práci vypracovala samostatně a použila jsem pouze podklady uvedené v přiloženém seznamu.

Declaration

I declare that I wrote my research work independently and exclusively with the use of cited bibliography.

Praha, 2013

Gabriela Malenová

Abstrakt

Název práce:
Spektra kvantových grafů

Autor: Gabriela Malenová

Obor: Matematické inženýrství

Druh práce: Diplomová práce

Vedoucí práce: Mgr. David Krejčiřík, DSc. ÚJF AV ČR, Řež

Abstrakt: Kvantový graf je struktura, která je determinována metrickým grafem, skládajícím se z množiny hran a vrcholů, diferenciálním operátorem, definovaným na hranách, podmínkami spojitosti ve vnitřních a hraničními podmínkami ve vnějších vrcholech. Jelikož můžeme určit spektrum kvantového grafu analyticky pouze v omezeném množství případů, je pro obecný graf zapotřebí numerických metod. Pro tento účel se jeví nejvhodnější spektrální metody odvozené od Galerkinovy tau-metody. Pro výpočet vlastních hodnot obecného grafu byl v prostředí MATLAB vyvinut numerický algoritmus. Tím získáme rozsáhlý soubor dat, který nám umožní pochopení základních vlastností kvantových grafů.
Zkoumána byla zejména spektrální mezera, neboli druhá vlastní hodnota standardního laplaciánu na metrickém grafu, a její vztah k algebraické konektivitě, především pak důsledky odebrání nebo přidání hrany do grafu. Přestože spolu některé vlastnosti kvantových a kombinatorických grafů korespondují, jejich vztah není vzájemně jednoznačný. Velikost spektrální mezery závisí nejen na topologii metrického grafu, ale také na jeho geometrických vlastnostech. Je dokázáno, že přidáním dostatečně dlouhé hrany nebo odebráním dostatečně krátké části hrany dosáhneme zmenšení spektrální mezery. V textu jsou zahrnuta i příslušná explicitní kritéria.
Další z důležitých výsledků říká, že řetízkový graf má vždy nejnižší spektrální mezeru mezi grafy stejné celkové délky.

Klíčová slova: kvantový graf, spektrální mezera, Rayleighův teorém, spektrální metody, Chebfun

Abstract

Title:
Spectra of Quantum Graphs

Author: Gabriela Malenová

Abstract: Quantum graph is a network structure determined by a metric graph consisting of sets of edges and vertices, a differential operator acting on the edges and matching and boundary conditions on internal and external vertices respectively. Since the spectra of quantum graphs can be calculated analytically in a few special cases only, numerical methods have to be employed. Spectral methods based on Galerkin tau-methods appear to be the most convenient for that purpose. The code in MATLAB environment has been evolved for computing eigenvalues of a general graph. Employing numerics, we obtain extensive computational data that may be helpful for understanding fine spectral properties of quantum graphs.
Above all, the spectral gap, i. e. the second eigenvalue of the standard Laplacian on metric graphs and the relation to the graph's algebraic connectivity has been closely investigated, in particular what happens to the gap if an edge is added to (or deleted from) a graph. In spite it bears some similar characteristic to discrete graphs the connection between the connectivity and the spectral gap is not one-to-one. The size of the spectral gap depends not only on the topology of the metric graph but on its geometric properties as well. It is shown that adding sufficiently large edges as well as cutting away sufficiently small edges leads to a decrease of the spectral gap. Corresponding explicit criteria are given. Another important result says that a string has always the lowest spectral gap among all graphs of the same total length.

Key words: quantum graph, spectral gap, spectral methods, Rayleigh theorem, Chebfun.

Contents

1 Introduction

A remarkable progress in the nanotechnology has been made in the last decades. It enabled one to exhibit quantum phenomena in the nanodevices because their typical length is comparable to the atom size. This raised the demand on mathematical studies of the quantum networks since they may be used to model such systems.

The origin of quantum graph theory may be traced back to 80's when the initial concept has been introduced (see [12] and the references therein). In the recent years, articles related to this topic were published on a regular basis as the concept gained enormous popularity. [11], [19], [18] are counted among the crucial papers. Furthermore, we refer to the survey [20]. The definitions are mainly taken from [21].

More specifically, quantum graphs consist of a *metric graph* Γ, i. e. linear network-shaped structure nesting set of edges $\mathbf{E}$ and vertices $\mathbf{V}$, a *differential operator* acting on the edges with *matching conditions* imposed at the vertices. An intuitive quantum graph model employs the standard Laplacian, i. e. Laplacian on $H^2(\Gamma\backslash\mathbf{V})$ satisfying the *standard matching conditions* in each vertex:

$$\begin{cases} \text{continuity of the functions} \\ \text{the sum of normal derivatives is zero.} \end{cases}$$

This guarantees that the Laplacian is self-adjoint on the graph Γ. More precise definition is provided in Section 2.

In this thesis, we consider compact quantum graphs. In general, it is not always possible to analytically find the spectrum since the number of explicitly solvable models is restricted. Some of them, e. g. the string, star, loop and lasso graphs, are presented in Section 3.

In more complicated cases, the numerical methods have to be applied. In Section 4, the spectral method approach is described that enables us to compute spectrum of a Schrödinger operator on an arbitrary quantum graph. Spectral methods based on the Chebyshev polynomials interpolation grant excellent accuracy.

Once having a tool computing the spectra in hand, we drew our attention to the inverse problems. As a first application, we computed the Euler characteristic derived from the *trace formula* in Subsection 5.1. This gives us the feeling about the number of terms in the sequence that are necessary for achieving requested accuracy.

However, the main objective of the current thesis is the *spectral gap*, the second eigenvalue of the standard Laplacian, in particular its relation to the graph's algebraic connectivity. As we carried out extensive numerical experiments on spectral gap (see Subsection 5.2), some theoretical observations and predictions have been formulated. Based on this proposals, theorems in Section 6 were proved that led to publications [24] and [23]. Our studies were inspired by classical results going back to Czechoslovak mathematician M. Fiedler [13] on the second eigenvalue of discrete graphs and by the recent paper by P. Exner and M. Jex on the ground state for quantum graphs with delta-coupling [10].

M. Fiedler proposed to call the second lowest (the first excited) eigenvalue of the discrete Laplacian the *algebraic connectivity*[1] of the corresponding discrete graph. This name proposal is explained by the close relation between algebraic connectivity and standard vertex and edge connectivities.

Recently, the spectral gap was also investigated numerically on large random graphs in [16]. The research concluded, vaguely said, that the algebraic connectivity may be taken as a measure of *synchronizability and robustness*. This found its application in neuron networks or signal transfer area.

P. Exner and M. Jex investigated the behavior of the ground state for (continuous) Laplacians on metric graphs as one of the edges is shortened or extended. It was shown that the bound state may increase as the length of an edge is increasing, however the opposite behavior may also occur.

Our goal is to study the behavior of the first excited eigenvalue when edges are either deleted or added to a metric graph. Bearing in mind that the ground state for standard Laplacian on the compact graph is zero, the first excited eigenvalue gives us the spectral gap (provided the graph is connected).

Spectral properties of the quantum graphs, especially with equilateral lengths of edges, are closely related to spectral properties of the corresponding discrete Laplacian. Therefore one might expect that the qualitative behavior of eigenvalues for discrete and continuous Laplacians is the same. However, it has been shown that the spectral gap for discrete and continuous Laplacians may behave differently as edges are added or deleted without altering the vertex set. This is connected to the fact that adding an edge to a discrete graph does not change the phase space, while adding an edge to a quantum graph enlarges the corresponding phase space.

Adding or deleting an edge without altering the vertex set has an influence on the graph's Euler characteristic. More precisely, it has been proven that the Euler characteristic is determined by spectral asymptotics and therefore can not be retrieved from the first few eigenvalues themselves, unless the metric graph consists of edges that are integer multiples of a basic length [22],[26].

It has been proven in [24] that the graph formed by just one edge (or a chain of edges) has the lowest spectral gap among all quantum graphs having the same total length $\mathcal{L}$. The proof is provided in Section 6.3. Therefore it is natural to expect that the spectral gap increases with the connectivity. On the other hand, adding an edge increases the total length as well and it might tempt one to expect the eigenvalues to drop according to Weyl's law as $\lambda_n \sim \left(\frac{\pi}{\mathcal{L}}\right)^2 n^2$. Taking into account these influences it turns out that both an increase and a decrease of the spectral gap are possible. We prove that unlike in the discrete case, the spectral gap is not granted to be monotonously dependent on the number of edges.

Note that the dependence of the graph's spectrum on the coupling constant at the vertices and the edge lengths has been also investigated in the interesting paper [2] (see also the recent book by the same authors [3]). A thorough analysis of quantum graphs and their approximations is also provided in the book by

[1] Whereas it is sometimes called the Fiedler value.

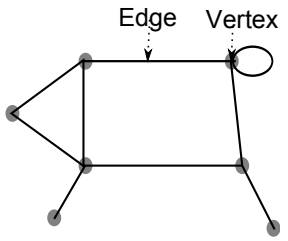


Figure 1: General metric graph.

Olaf Post [27].

2 Quantum graphs

Rigorous definition of the quantum graph contains three main parts:

1. the metric graph,
2. the differential operator acting on the edges,
3. the matching and boundary conditions at internal and external vertices respectively.

These conditions are not completely independent as will be explained below. The definitions are taken from the draft of the book by Pavel Kurasov [21].

2.1 Metric graph

In a broad sense, a metric graph is said to be a finite set of edges and vertices of given edge lengths (Figure 1). The edges may have finite or infinite length. More precisely, let us define the set $\{E_n\}_{n=1}^N$ of N compact or semi-infinite intervals E_n, each one of them being a subset of $\mathbb{R}$, as:

$$E_n = \begin{cases} [x_{2n-1}, x_{2n}], & n = 1, 2, \ldots, N_c \\ [x_{2n-1}, \infty), & n = N_c + 1, \ldots, N_c + N_i = N, \end{cases}$$

where N_c, respectively N_i, denotes the number of compact, respectively infinite, intervals. The intervals E_n are called edges.

Let us define the set $\mathbf{V}$ of all endpoints

$$\mathbf{V} = \{x_{2n-1}, x_{2n}\}_{n=1}^{N_c} \cup \{x_{2n-1}\}_{n=N_c+1}^{N},$$

and its arbitrary partition into M equivalence classes V_m, $m = 1, 2, \ldots, M$, called vertices. The equivalence classes have the following properties:

$$\mathbf{V} = V_1 \cup V_2 \cup \ldots \cup V_m$$
$$V_m \cap V_{m'} = \emptyset, \quad \text{when} \quad m \neq m'.$$

The endpoints belonging to the same equivalence class will be identified

$$x \sim y \Leftrightarrow \left[\begin{array}{l} \exists n : x, y \in E_n \ \& \ x = y, \\ \exists m : x, y \in V_m. \end{array} \right.$$

Definition 1. Let us have N edges E_n and a the set of M disjoint vertices V_m. Then the corresponding metric graph Γ is the union of all edges with the endpoints belonging to the same vertex identified

$$\Gamma = \bigcup_{n=1}^{N} E_n|_{x \sim y}.$$

The number v_m of elements in the class V_m will be called the *valence* of V_m.

We will mainly concentrate on compact graphs which occur when $N_i = 0$, i. e. all the edges are of finite length and $N = N_c$. Let us consider a complex-valued function u defined on the graph. Then the corresponding Hilbert space yields

$$L_2(\Gamma) = \bigoplus \sum_{j=1}^{N} L_2(E_j).$$

2.2 Differential operator

To properly implement dynamics of the waves on the graph, one introduces a differential operator. In general, magnetic Schrödinger operator

$$L_{q,a} = \left(i \frac{d}{dx} + a(x) \right)^2 + q(x), \tag{1}$$

is a standard choice for describing quantum phenomena, where a denotes the magnetic potential and q the electric potential respectively. More precisely, we assume $a(x), q(x) \in \mathbb{R}$ satisfying:

1. $q \in L_2(\Gamma)$,
2. $\int_\Gamma (1 + |x|) \cdot |q(x)| dx < \infty$,
3. $a \in C^1(\Gamma)$.

Let us take a function u belonging to the Sobolev space $H^2(E_n)$. Even if the endpoints coincide, one may set the boundary value of the function as a limit

$$u(x_j) = \lim_{x \to x_j} u(x).$$

For the boundary points, the *extended normal derivatives* are defined following the convention that the limits are taken in the direction pointing inside the respective interval:

$$\partial_n u(x_j) = \left\{ \begin{array}{ll} \lim_{x \to x_j} \left(\frac{d}{dx} - ia(x) \right) u(x), & x_j \quad \text{is the left end point,} \\ -\lim_{x \to x_j} \left(\frac{d}{dx} - ia(x) \right) u(x), & x_j \quad \text{is the right end point} \end{array} \right. . \tag{2}$$

Putting the magnetic potential in (1) equal to zero $a = 0$ we obtain Schrödinger operator

$$L_q = -\frac{d^2}{dx^2} + q(x). \tag{3}$$

Setting the potentials $a = 0 = q$ we get the Laplace operator describing the free motion:

$$L = -\frac{d^2}{dx^2}. \tag{4}$$

Then the normal derivatives (2) in the endpoints simplify to

$$\partial_n u(x_j) = \begin{cases} \lim_{x\to x_j} \frac{d}{dx} u(x), & x_j \quad \text{is the left end point,} \\ -\lim_{x\to x_j} \frac{d}{dx} u(x), & x_j \quad \text{is the right end point} \end{cases}. \tag{5}$$

Hereby we list some types of domains the Laplacian may be defined on. Firstly, let us consider the *maximal operator* $L^{\max}$ corresponding to (4) defined on the domain $D(L^{\max}) = H^2(\Gamma\backslash\mathbf{V})$, where H^2 denotes the Sobolev space of all square integrable functions having square integrable first and second derivatives. This domain may be written in the decomposed fashion as the sum of Sobolev spaces on the intervals E_n

$$D(L^{\max}) = \bigoplus \sum_{n=1}^{N} H^2(E_n),$$

independently on how the edges are connected to each other. Similarly, the operator $L^{\max}$ can be decomposed as

$$L^{\max} = \bigoplus \sum_{n=1}^{N} L^n,$$

where L^n is given by (4) on the domain $H^2(E_n)$.

Similar relations hold for the *minimal operator* $L^{\min}$ defined on $C_0^\infty(\Gamma\backslash\mathbf{V})$.

2.3 Matching conditions

The vertices may be divided into two groups. The first group is formed by *internal* vertices which have valence greater than one, in other words there are at least two edges meeting in the vertex. The subset of all conditions introduced at the internal points is called *matching conditions.* The other group is made up of vertices of valence equal to one called *boundary* vertices and *boundary conditions* are enforced there (see Figure 1).

The maximal operator $L^{\max}$ is neither self-adjoint nor symmetric. The self-adjointness may be achieved by imposing certain conditions on $u, v \in D(L^{\max})$:

$$\begin{aligned} \langle L^{\max}u, v\rangle - \langle u, L^{\max}v, \rangle &= \sum_{n=1}^{N} \left(\int_{E_n} -u''(x)\overline{v(x)}\,dx + \int_{E_n} u(x)\overline{v''(x)}\,dx \right) = \\ &= \sum_{x_j\in\mathbf{V}} \left(\partial_n u(x_j)\overline{v(x_j)} - u(x_j)\overline{\partial_n v(x_j)} \right), \end{aligned} \tag{6}$$

where the normal derivative at the endpoints is recalled from (5). Thus the operator $L^{\max}$ to be symmetric requires the boundary form in (6) being equal to 0.

What comes to one's mind at first is to require that the functions are equal to zero at the vertices.

Definition 2. The *Dirichlet Laplace operator* L^D is defined by the differential expression (4) on the Sobolev space $H^2(\Gamma\backslash\mathbf{V}) \hookrightarrow C^1(\Gamma\backslash V)$ satisfying the Dirichlet conditions

$$u(x_j) = 0,\ x_j \in \mathbf{V},$$

at all end points.

The Dirichlet Laplacian, may by presented in the decomposed way as:

$$L^D = \bigoplus \sum_{n=1}^{N} L^{n,D},$$

where $L^{n,D}$ is the differential operator (4) restricted to the set of all functions from the Sobolev space $H^2(E_n)$ satisfying the Dirichlet boundary conditions at endpoints. However, this is not an interesting case, since such an operator builds a graph model where all the edges are separated from each other and behave independently.

Another way how to impose conditions (6) without separating the edges is through the *standard matching and boundary conditions*[2] introduced in each vertex V_m:

$$\begin{cases} u \text{ is continuous at } V_m \\ \sum_{x_j \in V_m} \partial_n u(x_j) = 0. \end{cases} \tag{7}$$

For boundary vertices this simply yields the Neumann boundary condition

$$\partial_n u(x_j) = 0,\ x_j \in V_m,\ V_m \text{ is a boundary vertex.}$$

If there were two edges connected in a vertex this would imply nothing else than the continuity of the function and its first derivative. In that case, the vertex may be removed and the two intervals may be substituted by one of the sum of original sizes. The matching conditions give us a tool for setting up a self-adjoint operator called *standard Laplace operator* L^{st}:

Definition 3. The standard Laplace operator L^{st} is defined by the differential expression (4) on the domain $H^2(\Gamma\backslash\mathbf{V})$ satisfying the standard matching conditions (7) at all vertices.

Since the standard Laplace operator is self-adjoint, is possesses real spectrum.

[2]Standard matching conditions are sometimes called *Free, Neumann or Kirchhoff conditions.*

2.4 Elementary spectral properties

Since we consider compact graphs formed by finitely many edges, spectral properties may be characterized by the following theorem.

Theorem 2.1 ([21]). *Let Γ be a finite compact graph and $L_q = L^{\mathrm{st}} + q$ the corresponding Schrödinger operator* (3)*. Then the spectrum is purely discrete and consists of infinite sequence of real eigenvalues with one accumulation point $+\infty$.*

The proof may be found in [21]. Note, that the standard Laplacian as well as the Dirichlet Laplacian are in fact the extensions of the symmetric operator defined by the same formula (4) on the domain of all continuous functions from $H^2(\Gamma\backslash\mathbf{V})$ subject to the following conditions:

$$\begin{cases} u(V_m) = 0 \\ \sum_{x_j \in V_m} \partial u(x_j) = 0 \end{cases},$$

for all vertices.

Some important facts regarding the standard Laplacian remain to be proven. Above all, the first eigenvalue of the compact graph is always equal to zero.

Proposition 2.2. *$\lambda_0 = 0$ is the first eigenvalue of the standard Laplace operator L^{st} on finite compact graph Γ with multiplicity n equal to number of connected components. The corresponding eigenvector is equal to $1 \in L_2(\Gamma_i)$ on the i-th component and zero elsewhere.*

Proof. The eigenvectors corresponding to $\lambda_0 = 0$ are linear functions since they satisfy

$$-\psi'' = 0, \qquad \psi(x) = \alpha_n x + \beta_n,$$

on each edge E_n. Application of the standard matching conditions (7) preserves continuity of the eigenvectors which makes their maxima attainable on the endpoints only. Say, we achieved the global maximum. Sum of the derivatives is zero at each node, but in this point, they have to be non-positive (due to the maximum). This necessarily implies, they are identically equal to zero. All in all, ψ is constant on every edge attached to the maximum. We conclude that the function is constant on all such edges. Consider another neighboring vertex and repeat the arguments. We continue in this way until the whole connected component is covered. This brings us to the claim, that the spectral multiplicity of the eigenvalue $\lambda_0 = 0$ is the number of connected components. □

However, it is not always necessary to compute the whole infinite sequence of eigenvalues. For example if the lengths of all edges are integer multiples of a basic length then the spectrum is periodic and it is enough to calculate just the first few eigenvalues to know the whole spectrum.

Proposition 2.3 ([21]). *Let $k^2 \neq 0$ is an eigenvalue of the Schrödinger operator L^{st}* (4) *and the quadratic form* (6) *is equal to zero on a graph Γ formed by edges having basic length $\Delta : \ell_j = m_j\Delta$, $m_j \in \mathbb{N}$, then $(k + \frac{2\pi}{\Delta})^2$ also belongs to the spectrum.*

Proof. Let us consider $k^2 \in \sigma(L^{\mathrm{st}}(\Gamma))$. Then the eigenvector ψ restricted to the n-th edge $[x_{2n-1}, x_{2n}]$ is given by (for the derivation see [21]):

$$\psi(x)|_{E_n} = a_{2n-1}e^{ik|x-x_{2n-1}|} + a_{2n}e^{ik|x-x_{2n}|}.$$

Shifting the frequency

$$k \longrightarrow k + \frac{2\pi}{\Delta},$$

we obtain a new function $\widetilde{\psi}$:

$$\widetilde{\psi}(x)|_{E_n} = a_{2n-1}e^{i(k+\frac{2\pi}{\Delta})|x-x_{2n-1}|} + a_{2n}e^{i(k+\frac{2\pi}{\Delta})|x-x_{2n}|}.$$

Comparing the boundary values we get

$$\widetilde{\psi}(x_{2n}) = a_{2n-1}e^{ikm_n\Delta} + a_{2n} = \psi(x_{2n}).$$

Similarly,

$$\widetilde{\psi}(x_{2n-1}) = \psi(x_{2n-1}).$$

Analogously, the derivatives are carried out:

$$\widetilde{\psi}'(x_{2n-1}) = i\left(k + \frac{2\pi}{\Delta}\right)(a_{2n-1} + a_{2n}e^{ikm_n\Delta}) = \left(1 + \frac{2\pi}{\Delta k}\right)\psi'(x_{2n-1}),$$

and

$$\widetilde{\psi}'(x_{2n}) = \left(1 + \frac{2\pi}{\Delta k}\right)\psi'(x_{2n}).$$

This means, $\widetilde{\psi}$ is the eigenfunction corresponding to the eigenvalue $\left(k + \frac{2\pi}{\Delta}\right)^2$. Therefore the matching conditions are satisfied. □

3 Explicit solutions

Let us first introduce some elementary cases of quantum graphs where the spectrum can be calculated explicitly. The most natural starting point is to consider the Laplacian on a single interval.

3.1 Interval

We have the Dirichlet Laplacian (Def 2) on a single interval $[x_1, x_2]$. Consequently, we may reparametrize it to $[0, l]$. Thus we are to solve the problem

$$\begin{cases} -u'' = \lambda u \\ u(0) = 0,\ u(l) = 0. \end{cases}$$

Any solution to the differential equation can be obtained in the form

$$u(x) = A\cos kx + B\sin kx, \quad A, B \in \mathbb{C}, \tag{8}$$

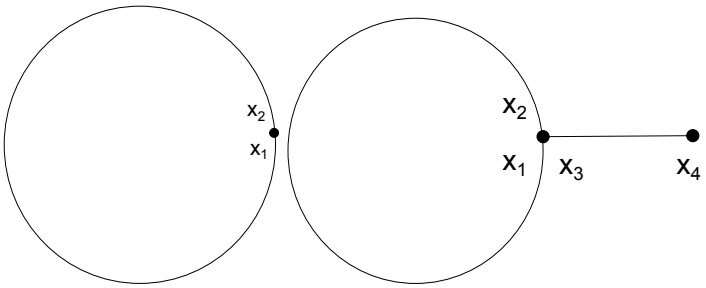


Figure 2: Loop and lasso graphs.

where $k^2 = \lambda$ which implies the eigenvalues being in the form of infinite sequence

$$\lambda_n = \frac{\pi^2}{l^2} n^2, \quad n = 1, 2, \ldots, \tag{9}$$

with the eigenvectors

$$u(x) = \sin \frac{\pi n x}{l}, \quad n = 1, 2, \ldots.$$

Similar calculations can be carried out for the standard operator (Def 3) on the same interval. Thus we need to solve the problem given by

$$\begin{cases} -u'' = k^2 u \\ u'(0) = 0,\ u'(l) = 0, \end{cases} \tag{10}$$

where the solution form is recalled from (8). Then the constraint to be solved is

$$k \sin kl = 0,$$

whose solution is the sequence

$$\lambda_n = \frac{\pi^2}{l^2} n^2, \quad n = 0, 1, 2, \ldots \tag{11}$$

as before, with the eigenvectors

$$u(x) = \cos \frac{\pi n x}{l}, \quad n = 0, 1, 2, \ldots.$$

Notice, that in addition there is the zero eigenvalue $\lambda_0 = 0$ included as well, with the eigenvector $u(x) = 1$.

3.2 Loop graph

Another case of a graph formed by just one edge is a *loop* (Figure 2, left). Here, the endpoints are identified and the edge consists of one interval $[x_1, x_2] = [0, l]$. The stationary Schrödinger equation yields

$$-u'' = k^2 u,$$

where $\lambda = k^2$ is the eigenvalue of the differential operator L^{st}. Matching conditions imply that

$$\begin{cases} u(0) = u(l) \\ u'(0) = u'(l) \end{cases}.$$

Plugging this into Ansatz (8) we arrive at the constraint

$$2k(1 - \cos kl) = 0.$$

The eigenvalue $\lambda = 0$ is a simple eigenvalue with the eigenfunction $u = 1$. The eigenvalues

$$\lambda_n = \frac{4\pi^2}{l^2} n^2, \quad n = 1, 2, \ldots, \tag{12}$$

are of the multiplicity 2. The eigenvectors may be split into two groups according to criterion whether they are or are not invariant under the change of variables $x \mapsto l - x$. Then the even, respectively odd functions are denoted by u_e, respectively u_o, and they satisfy

$$u_e(x) = \cos \frac{2\pi}{l} nx, \quad u_o(x) = \sin \frac{2\pi}{l} nx.$$

3.3 Lasso graph

The obvious way to proceed further is to start connecting the intervals together. The *lasso graph* (Figure 2, right) is built up by attaching an interval to a loop. Mathematically, this graph Γ may be defined as a union of two intervals $E_1 = [x_1, x_2]$ and $E_2 = [x_3, x_4]$ with the endpoints x_1, x_2, x_3 identified in one vertex $V_1 = \{x_1, x_2, x_3\}$. In the view of symmetry, it is more convenient to choose the parametrization of edges as follows:

$$[x_1, x_2] = [-l/2, l/2], \quad [x_3, x_4] = [0, L].$$

The operator is invariant under the change of variables

$$J : x \mapsto \begin{cases} -x, & x \in E_1, \\ x, & x \in E_2. \end{cases}$$

This transformation can be lifted to act on functions

$$(Jf)(x) = f(Jx).$$

We see that the Laplacian is commuting with J

$$JL = LJ,$$

hence the corresponding Laplacian eigenfunctions may be chosen symmetric and antisymmetric with respect to J:

$$Jf_{\mathrm{sym}} = f_{\mathrm{sym}}, \quad Jf_{\mathrm{asym}} = -f_{\mathrm{asym}}.$$

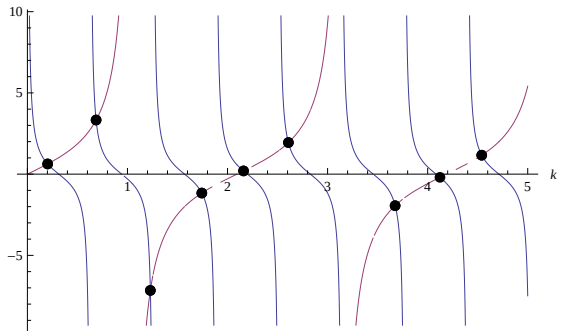


Figure 3: Graphical solution of equation (15) for $L = 5$ and $l = 3$.

The Laplacian is self-adjoint when defined on functions satisfying the following conditions

$$\begin{cases} u(x_4) = 0, \\ u(x_1) = u(x_2) = u(x_3) \\ u'(x_1) - u'(x_2) + u'(x_3) = 0. \end{cases} \tag{13}$$

Let us first start with the antisymmetric functions. They are necessarily equal to zero on the second interval. On the loop, the eigenfunctions are of the form $u(x) = A \sin kx$ due to the antisymmetry. This requires zero value in the middle point, i. e. the condition

$$A \sin kl/2 = 0, \tag{14}$$

which is satisfied if

$$\lambda_n = \frac{4\pi^2 n^2}{l^2}, \quad n = 1, 2, \dots.$$

The third condition in (13) obviously holds due to antisymmetry. The symmetric eigenfunctions are of the form

$$u = \begin{cases} D \cos kx, & x \in E_1 \\ C \sin k(x - L), & x \in E_2. \end{cases}$$

To satisfy the second condition in (13), the following constraint has to hold true:

$$D \cos kl/2 = C \sin(-kL).$$

The third condition in (13) implies the equation:

$$2D \sin kl/2 + C \cos kL = 0.$$

The two equations form a 2×2 linear system which has a nontrivial solution if and only if the corresponding determinant is equal to zero:

$$\cot kL = 2 \tan kl/2. \tag{15}$$

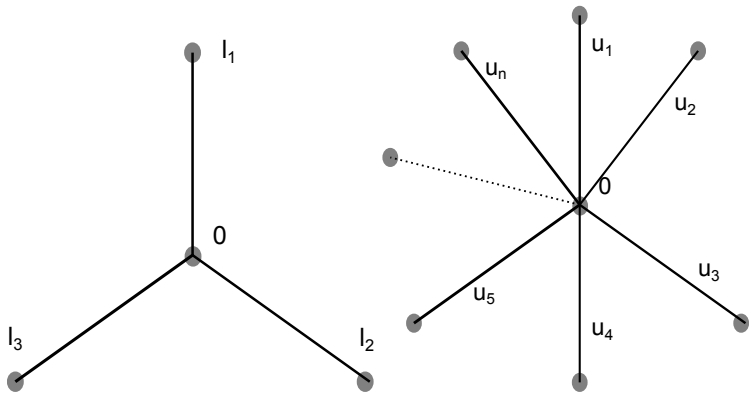


Figure 4: 3-star and n-star graph.

The graphical solution for cases $L = 5$ and $l = 3$ is depicted in Figure 3. Finally, joining symmetric and antisymmetric constraints (14) one obtains the eigenvalues condition:

$$(\cot kL - 2\tan kl/2)\sin kl/2 = 0.$$

Thus the solution can be computed explicitly only in case L and l are rationally dependent.

3.4 3-star graph

Let us consider a star-shaped graph, namely a set of edges of arbitrary lengths meeting in one central point (see Figure 4, left). The three edges' lengths are denoted by l_1, l_2 and l_3 respectively. The most convenient way of parametrization is to design each edge as $E_n = [0, l_n]$. Thus the solution (applying standard matching conditions) satisfies the following system of equations:

$$\begin{cases} u_1(0) = u_2(0) = u_3(0), \\ u_1'(0) + u_2'(0) + u_3'(0) = 0, \\ u_1'(l_1) = u_2'(l_2) = u_3'(l_3) = 0, \end{cases}$$

where u_j denotes the values of the function u on one of the three intervals. The functions u_j are of the form (8):

$$u_j(x) = A_j \cos k(x - l_j) + B_j \sin k(x - l_j).$$

Applying the conditions mentioned just above, we end up with the matrix equation

$$\underbrace{\begin{pmatrix} \cos kl_1 & -\cos kl_2 & 0 \\ 0 & \cos kl_2 & -\cos kl_3 \\ \sin kl_1 & \sin kl_2 & \sin kl_3 \end{pmatrix}}_{=:M} \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} = \hat{0}$$

The requirement of the solution to be non-trivial leads to the condition that the determinant of the matrix M is zero:

$$\det M = 0.$$

The equation may be re-written as

$$\begin{aligned} 0 &= \cos kl_1 \cos kl_2 \sin kl_3 + \cos kl_1 \sin kl_2 \cos kl_3 + \sin kl_1 \cos kl_2 \cos kl_3 = \\ &= \cos kl_2 \sin k(l_1 + l_3) + \frac{1}{2} \sin kl_2 [\cos k(l_1 - l_3) + \cos k(l_1 + l_3)], \end{aligned} \tag{16}$$

or similarly, after some algebra:

$$0 = 3 \sin kL + \sin k(-l_1 + l_2 + l_3) + \sin k(l_1 - l_2 + l_3) + \sin k(l_1 + l_2 - l_3),$$

where $L = l_1 + l_2 + l_3$. Solutions to this equation may be computed numerically.

Special case of two edges of the star graph having the same length was considered. Here on we set $l_1 = l_3 = l$, which brings us to the constraint

$$\begin{aligned} 0 &= \cos kl(2 \cos kl_2 \sin kl + \sin kl_2 \cos kl) = \\ &= \cos kl(\cos kl_2 \sin kl + \sin k(l + l_2)). \end{aligned}$$

This implies two types of solution. First we have

$$\cos kl = 0 \qquad \Longrightarrow \qquad k_n = \frac{\pi}{l} \left(\frac{1}{2} + n \right), \; n = 0, 1, \ldots. \tag{17}$$

The other equation has to be evaluated numerically, k is the solution of the following equation:

$$0 = 2 \cos kl_2 \sin kl + \sin kl_2 \cos kl. \tag{18}$$

The solutions of the above equations coincide only if there are some $n, m \in \mathbb{Z}$ such that

$$\frac{l_2 - l}{2} = ml - nl_2.$$

3.5 Equilateral star graph

In general, a star graph may be built up from a higher number of branches. The computations, as the number rises, are getting excessively large. The only case we are able to solve the problem analytically occurs when all the edges have the same length l.

Let us start with an n-star graph presented in Figure 4. Taking standard matching conditions into account, we require

$$\begin{cases} u_1(0) = u_2(0) = \ldots = u_n(0), \\ \sum_i u_i'(0) = 0, \\ u_1'(l) = u_2'(l) = \ldots = u_n'(l) = 0. \end{cases} \tag{19}$$

We take advantage of the graph being rotationally symmetric with respect to the central node. Let us define the operator of rotation R as

$$R(u_1, u_2, u_3, \dots, u_n) := (u_2, u_3, \dots, u_n, u_1).$$

The operators L^{st} and R commute

$$RL^{\mathrm{st}} = L^{\mathrm{st}}R.$$

This leads to the eigenvalue problem

$$L^{\mathrm{st}}u = \lambda u, \qquad \text{where } Ru = \mu u,$$

with u being the eigenvector of both L^{st} and R since they are self-adjoint.

Due to the fact

$$R^n = 1$$

eigenvalues of R are n-th roots of 1. As we already mentioned, the threshold eigenvalue λ_0 of a standard Laplacian is always 0 and the corresponding eigenvector is $1 \in L_2(\Gamma)$. This is the case of μ_0 equal to one:

$$R(1, 1, 1, \dots, 1) = 1 \cdot (1, 1, 1, \dots, 1).$$

The eigenvector corresponding to $\mu_1 = e^{i\frac{2\pi}{n}}$ obeys

$$R\left(1, e^{i\frac{2\pi}{n}}, e^{2i\frac{2\pi}{n}}, \dots, e^{(n-1)i\frac{2\pi}{n}}\right) = \left(e^{i\frac{2\pi}{n}}, e^{2i\frac{2\pi}{n}}, \dots, e^{(n-1)i\frac{2\pi}{n}}, 1\right) = \\ = e^{i\frac{2\pi}{n}}\left(1, e^{i\frac{2\pi}{n}}, e^{2i\frac{2\pi}{n}}, \dots, e^{(n-1)i\frac{2\pi}{n}}\right).$$

Similarly, we may proceed further by analogously applying multiple rotations on the vector, henceforth we arrive at

$$\mu_k = e^{ki\frac{2\pi}{n}}$$

and the respective eigenspaces read as follows:

$$\begin{aligned} (1, z, z^2, \dots z^{n-1}) \times L_2([0, l]), \\ (1, z^2, z^4, \dots z^{2(n-1)}) \times L_2([0, l]), \\ \vdots \end{aligned}$$

providing $z = e^{i\frac{2\pi}{n}}$. It means that one can look for eigenfunctions of L of the form

$$Rf = z^k f, \quad k = 0, 1, \dots n-1.$$

Setting $k = 0$ gives us symmetric functions while $k = 1, 2, \dots n-1$ defines quasi invariant functions. Functions from the latter class satisfy the standard matching conditions only if they are zero at the central vertex. Indeed, from the continuity condition in (66), it follows that

$$u_1(0) = \underbrace{e^{i\frac{2\pi}{n}}u_1(0)}_{u_2(0)} \qquad \Longrightarrow \quad u_1(0) = 0,$$

since $e^{i\frac{2\pi}{n}} \neq 1$.

The condition on the derivative in (66) is satisfied due to quasi invariance, indeed:

$$u_1'(0) + u_2'(0) + \ldots + u_n'(0) = \left(1 + e^{i\frac{2\pi}{n}} + e^{2i\frac{2\pi}{n}} + \ldots + e^{(n-1)i\frac{2\pi}{n}}\right) u_1'(0) =$$
$$= \left(\frac{1 - e^{ni\frac{2\pi}{n}}}{1 - e^{i\frac{2\pi}{n}}}\right) u_1'(0) = 0,$$

and the same holds for its powers. Hence, we have

$$\begin{cases} u_1(0) = 0, \\ u_1'(l) = 0. \end{cases}$$

The solution is the sinus function

$$u_1(x) = B \sin kx,$$

which after some algebra implies

$$k_n = \frac{\pi}{2l} + \frac{n\pi}{l},$$

whose multiplicity is $n - 1$.

Let us proceed further with the symmetric part. For the conditions on the derivative in (66) to be satisfied we need

$$0 = \sum_i u_i'(0) = n u_1'(0),$$

which implies the solution in the form

$$\begin{cases} u_1'(0) = 0, \\ u_1'(l) = 0. \end{cases}$$

This is satisfied by the function

$$u_1(x) = A \cos k_n x.$$

if

$$k_n = \frac{\pi n}{l}.$$

Multiplicity of such an eigenvalue is 1.

4 Numerical analysis

In the preceding chapter, we have presented some of the explicitly solvable (or nearly explicitly solvable, by transforming into root-finding task) eigenvalue problems. However, their number is very limited. The problem gets excessively arduous when adding electric or magnetic potential. In further investigation, numerical computation plays an important role. The question arises which numerical method to choose to compute the spectrum of a general quantum graph endowed with Schrödinger operator and various boundary conditions. For reasons to be explained below, we use Chebyshev spectral methods and object-oriented MATLAB environment.

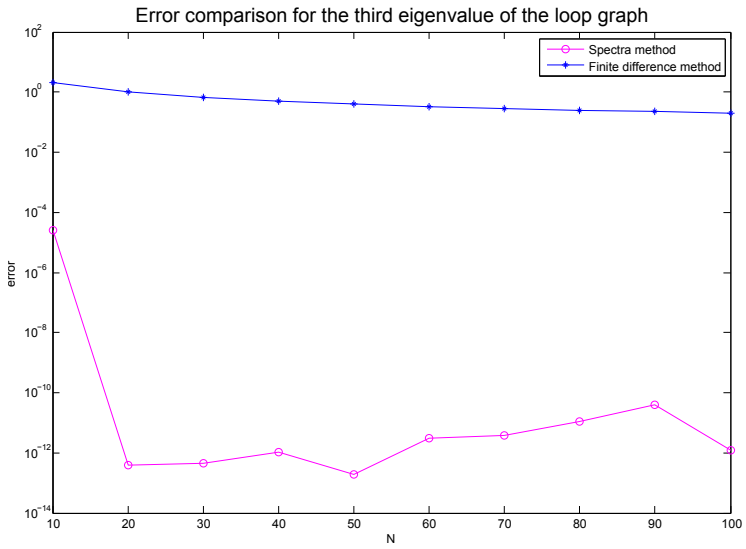


Figure 5: Compare the accuracy rate for spectral method and finite difference method of the first order of the eigenvalue λ_3 in the loop case (where the value is exactly known).

4.1 Chebyshev spectral methods

The first method that immediately comes to one's mind is some kind of finite difference formula. MATLAB takes use of sparse matrices, so the codes run in the fraction of seconds. However, the speed of such computation is at the expense of accuracy. From this point of view, spectral methods are more suitable for problems requiring high order of precision. In broad terms, while finite difference methods make use of local interpolation by low degree polynomials, spectral methods implement high degree polynomials globally. Spectral accuracy is remarkable, however, there is a price to be paid: full matrices replace sparse matrices, stability restrictions may become more severe, and computer implementations may not be so straightforward.

As the number of grid points N increases, the error for finite difference and finite element scheme typically decreases like $O(N^{-m})$ for some constant m depending on the order of approximation and the smoothness of the solution. For the spectral method, convergence of the rate $O(N^{-m})$ for *arbitrary* m is achieved, provided the solution is infinitely differentiable, and even faster convergence at a rate $O(c^N)$, $0 < c < 1$ is achieved if the solution is analytic [28].

This behavior is illustrated by Figure 5. The error of spectral and finite difference methods is plotted in the case of loop graph, where the solution (12) is explicitly known. Obviously, finite difference method result is improved very slowly compared to spectral method. Reaching $N = 20$ interpolation points,

spectral methods achieve the accuracy 10^{-12} where the truncation error does not allow the error to drop more. This is the typical spectral accuracy behavior.

4.1.1 Chebyshev nodes

The eigenvectors of Schrödinger operator (1) are smooth functions, thus according to [28] it is customary to interpolate them by algebraic polynomials $p(x) = a_0 + a_1 x + \dots a_n x^N$. To avoid Runge phenomenon (oscillations near the endpoints) it is convenient not to interpolate the function on equispaced points but to introduce the discretization on unevenly spaced points.

Various different sets of points are effective but they shall be distributed asymptotically as $N \to \infty$ with the density per unit length as

$$\text{density } \sim \frac{N}{\pi\sqrt{1-x^2}},$$

which means that they cluster near the endpoints. The canonical interval is $[-1, 1]$. If a function is to be evaluated on general interval $[a, b]$, a mapping comes handy when converting to $[-1, 1]$ through the change of variables

$$x \mapsto \frac{(b-a)x + (b+a)}{2}.$$

One of the node sets satisfying the density property on the bounded interval are *Chebyshev points.* There exist more types of them, the most commonly used ones as they are nearly optimal [4] are so called *Chebyshev Gauss-Lobatto quadrature points* [5]:

$$x_j = \cos\frac{j\pi}{N}, \quad j = 0, 1, \dots, N. \tag{20}$$

In the literature, the *Chebyshev points of the second kind* are sometimes used too:

$$x_j = \cos\frac{(j-1)\pi}{N-1}, \quad j = 1, 2, \dots, N. \tag{21}$$

Note, that the ordering is defined from right to left.

4.1.2 Chebyshev polynomials

Chebyshev points (20) are the roots of *Chebyshev polynomials of the first kind*, $T_k(x)$ where $k = 0, 1, \dots$, see Figure 6. Similarly, Chebyshev nodes (21) are the roots of $T_{k-1}(x)$ on $[-1, 1]$. In fact, Chebyshev polynomials are the eigenfunctions of the Sturm-Liouville problem

$$\left(\sqrt{1-x^2}\,T_k'(x)\right)' + \frac{k^2}{\sqrt{1-x^2}}\,T_k(x) = 0.$$

The polynomials may be also given by the recursion relation

$$T_{k+1}(x) = 2xT_k(x) - T_{k-1}(x), \quad T_0(x) = 1, \quad T_1(x) = x.$$

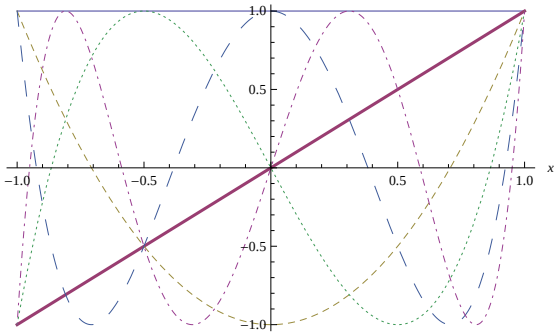


Figure 6: First six Chebyshev polynomials.

For more details see [5].

Chebyshev polynomials are real and orthogonal with respect to the weight $w(x) = \frac{1}{\sqrt{1-x^2}}$ on $(-1, 1)$

$$\int_{-1}^{1} T_n(x)\, T_m(x)\, \frac{dx}{\sqrt{1-x^2}} = \begin{cases} 0, & n \neq m, \\ \pi, & n = m = 0, \\ \pi/2, & n = m \neq 0, \end{cases}$$

and build the basis in the weighted space $L^2_w(-1, 1)$. Chebyshev expansion of a function $u \in L^2_w(-1, 1)$ is

$$u = \sum_{k=0}^{\infty} \hat{u}_k T_k(x), \quad \hat{u}_k = \frac{2}{\pi c_k} \int_{-1}^{1} u(x) T_k(x) w(x)\, dx,$$

where

$$c_k = \begin{cases} 2, & k = 0, \\ 1, & k \geq 1. \end{cases}$$

4.1.3 Differentiation matrix

There are two options for to accomplish differentiation of a function depending on its representation, we can either stay in the transform space $L^2_w(-1, 1)$ or express the function in physical space $L^2(-1, 1)$. The other way is preferred.

To carry out the computation in the physical space one needs to define its base first. *Characteristic Lagrange polynomials* ψ_l are natural choice- they are unique polynomials that satisfy

$$\psi_l(x_j) = \delta_{jl}, \quad j = 0, \ldots, N.$$

The general expression for such polynomials is

$$\psi_l(x) = \prod_{j \neq l, 0 \leq j, l \leq N} \frac{x - x_j}{x_l - x_j}. \tag{22}$$

For numerical stability reasons, often the Lagrangian polynomials are reformulated in *barycentric form* as

$$\psi_l(x) = \frac{\frac{\lambda_l}{x-x_l}}{\sum_{k=0}^{N} \frac{\lambda_k}{x-x_k}}, \qquad \lambda_l = \frac{1}{\prod_{k\neq l}(x_l - x_k)}. \tag{23}$$

Differentiation in physical space is realized by replacing truncation by interpolation. Given a set of $N+1$ nodes in $[-1,1]$ the polynomial

$$D_N u = \left(\sum_{l=0}^{N} u(x_l)\psi_l \right)'$$

is called the *Jacobi interpolation derivative* of u. The coefficients are given by $(D_N)_{jl} = \psi_l'(x_j)$, they form the entries of the *first-derivative interpolation matrix* D_N.

In our case it may be shown, that the characteristic Lagrange polynomials (22) at the Chebyshev Gauss-Lobatto points (20) may be expressed as

$$\psi_l(x) = \frac{(-1)^{l+1}(1-x^2)\, T_N'(x)}{\bar{c}_l N^2 (x - x_l)},$$

where

$$\bar{c}_j = \begin{cases} 2, & j = 0, N, \\ 1, & j = 1, \ldots, N-1. \end{cases}$$

From this, one gets the derivative interpolation matrix:

$$(D_N)_{jl} = \begin{cases} \frac{\bar{c}_j (-1)^{j+l}}{\bar{c}_l (x_j - x_l)}, & j \neq l, \\ -\frac{x_l}{2(1-x_l^2)}, & 1 \leq j = l \leq N-1, \\ \frac{2N^2+1}{6}, & j = l = 0, \\ -\frac{2N^2+1}{6}, & j = l = N. \end{cases}$$

Numerically more stable code takes advantage of the barycentric formula (23):

$$(D_N)_{jl} = \begin{cases} \frac{\delta_j}{\delta_l} \frac{(-1)^{j+l}}{x_j - x_l}, & j \neq l, \\ -\sum_{i=0, i\neq j}^{N} \frac{\delta_i}{\delta_j} \frac{(-1)^{i+j}}{x_j - x_i}, & j = l, \end{cases} \tag{24}$$

where $\delta_l = 1/2$ if $l = 0$ or N, $\delta_l = 1$ otherwise.

The ready-made function `chebdif.m` by Weideman and Reddy implements the expression similar to (24) [3] on Chebyshev nodes of the second kind (21). The documentation to the MATLAB suite may be found in [29]. The program makes use of the fact, that for spectral differentiation matrices it holds

$$D_N^{(l)} = (D_N^{(1)})^l, \quad l = 1, 2, \ldots, \tag{25}$$

thus any higher order differentiation matrix can be computed from (24).

[3] The spectral Chebyshev matrix is also included in the MATLAB's Matrix Computation Toolbox under `chebspec` label.

4.1.4 Eigenvalue problem and boundary conditions

Let us concentrate on the eigenvalue problem of the Laplace operator. In the discretized way, we are to solve:

$$-D_N^{(2)}u = \lambda u, \quad u = [u_0, \dots u_N]^T, \tag{26}$$

where $u_i := u(x_i)$ and $D_N^{(2)}$ is the second order differentiation matrix (25). In (26) we did not take into account boundary and matching conditions yet.

For finding the eigenvalues, MATLAB's command `eig` is a very powerful tool. Note, that by calling it, finite set of eigenvalues is returned. However, the accuracy decreases with the number of the eigenvalue. Quantum graphs have infinitely many eigenvalues and in order to get many one needs to consider large N. In the case of a graph with rationally dependent lengths of edges we recall that the eigenvalues are repeating within certain period (as proved in Proposition 2.3).

It might happen that one is interested in merely few first eigenvalues or would be working with extensive matrices where the computations are not in the power of modern computers. Iteration methods may solve the problem. One of the most frequently used techniques for computing a few dominant eigenvalues is the Lanczos algorithm [25]. However, in our thesis we work with adequately small matrices and smooth solutions, so the `eig` command is customary enough for our purpose.

For the Laplace operator to be self-adjoint, one needs to endow the boundary conditions. In general, there are two different approaches of implementing the boundary conditions for spectral methods:

1. restrict the set of interpolants to those, that satisfy the boundary conditions

2. do not restrict, but add additional equations to enforce the boundary conditions.

The first method is based on changing the form of interpolant basis to the so-called *boundary adapted* form. For theoretical background read the paper by Huang and Sloan [17], where the general form of the interpolant is provided. This method has been incorporated into the program `cheb2bc.m` by Weideman and Reddy, for more information see [29].

The second method is more flexible and suitable for more complicated problems. It is related to the so-called *tau methods* that appear in the field of Galerkin spectral methods. The boundary conditions are imposed in standard ways as described in Chapters 7 and 13 of [28]. Each boundary condition at the left endpoint modifies the initial row of the differentiation matrix, and each boundary condition on the right endpoint involves the modification in the final row.

Let us demonstrate the boundary condition implementation on the interval $[-1, 1]$. In the literature, the authors mainly concern about the Dirichlet boundary conditions $u(\pm 1) = c_{1,2}$ since they are easy to implement. Given $c_{1,2} = 0$,

which is the most commonly considered case, according to [28], this may be achieved by omitting the outer rows and columns of the differentiation matrix and adjusting the length of vector u by setting $u_0 = u_N = 0$.

To present the eigenvalue problem with Neumann boundary conditions (10), let us fist recall the differentiation matrix of the second order (25). Then the discretized problem yields

$$\begin{cases} -D_N^{(2)} u = \lambda u, \\ u_0' = u_N' = 0, \end{cases}$$

where $u_i' = (D_N^{(1)} u)_i$ is the i-th element of the differentiated vector u. In other words, we impose $N-1$ equations using the second order differentiation matrix and 2 equations using the first order differentiation matrix. So we will end up solving $(N+1)\times(N+1)$ linear system of equations where $N-1$ equation enforce the condition $-u'' = \lambda u$ at the interior grid and 2 equations enforce $u' = 0$ at the outer grid points:

$$-Au = \lambda B u, \tag{27}$$

where A is the modified differentiation matrix $D_N^{(2)}$. The technique of implementing Neumann boundary condition consists of replacing the first row in $D_N^{(2)}$ by the first row in $D_N^{(1)}$, respectively last row in $D_N^{(2)}$ by the last row in $D_N^{(1)}$. The matrix B is singular

$$B = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 0 & 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 \end{pmatrix},$$

thus this is a generalized eigenvalue problem, which may be solved by the MATLAB command `eig(A,B)`. Similarly, we enforce the non-homogeneous Dirichlet conditions.

4.2 Chebfun

There exist more add-ons for MATLAB implementing the techniques described above, see for instance [29]. The most complex toolbox utilizing Chebyshev spectral methods is *Chebfun*- an ongoing Oxford University project supervised by Lloyd Nick Trefethen that goes back to 2002 [1]. Chebfun is an open source package extending MATLAB environment providing one may conduct the computations in *functional form* (instead of vectorized form). The goal of Chebfun is to solve the problems with "symbolic feel and numerical speed". It is designed to aim on users which are already familiar with MATLAB as overloading the commands into Chebfun language offers completely new

view on their functionality. The current version of Chebfun is available at http://www2.maths.ox.ac.uk/chebfun/.

As was already mentioned in the introduction, a convenient advantage of spectral methods is that the error decreases rapidly as the number of interpolation points grows. One of the Chebfun features is the adaptivity- the number of polynomials necessary is computed during the process and their number aims to achieve the error drops to the magnitude of machine precision ($\sim e^{-16}$).

When an operator in Chebfun is constructed, the aim is always accuracy close to machine epsilon, however, because of the ill-conditioning associated with spectral discretizations, accuracy lost is almost universal [8]. As the examples in this thesis show, it is common to lose three or four digits of accuracy in such computations.

4.2.1 Current features

To present the ease of use for the spectral discretization described in the previous section, we employ Chebfun for computing the differentiation matrices. For example, the second order differentiation matrix evaluated at 5 points on the interval $[-1, 1]$ without boundary conditions can be obtained by command

```
d = domain(-1,1);
D2 = diff(d,2);
D2(5)
```

```
ans =

   17.0000  -28.4853   18.0000  -11.5147    5.0000
    9.2426  -14.0000    6.0000   -2.0000    0.7574
   -1.0000    4.0000   -6.0000    4.0000   -1.0000
    0.7574   -2.0000    6.0000  -14.0000    9.2426
    5.0000  -11.5147   18.0000  -28.4853   17.0000
```

If we want to apply Dirichlet boundary conditions, we simply call the below command. Note, that corresponding identity rows replace the first and last row[4].

```
D2.bc = 'dirichlet';
D2(5)
```

```
ans =
```

[4] In the future release of Chebfun, the so called rectangular projection is about to be implemented. This maps the interior rows to slightly different set of points as the computation coefficients appear to be more stable.

```
    1.0000         0         0         0         0
    9.2426  -14.0000    6.0000   -2.0000    0.7574
   -1.0000    4.0000   -6.0000    4.0000   -1.0000
    0.7574   -2.0000    6.0000  -14.0000    9.2426
         0         0         0         0    1.0000
```

With Neumann boundary conditions, the first and last rows are replaced by corresponding rows of the first derivative operator:

```
D2.bc = 'neumann'
D2(5)
```

```
ans =

   -5.5000    6.8284   -2.0000    1.1716   -0.5000
    9.2426  -14.0000    6.0000   -2.0000    0.7574
   -1.0000    4.0000   -6.0000    4.0000   -1.0000
    0.7574   -2.0000    6.0000  -14.0000    9.2426
    0.5000   -1.1716    2.0000   -6.8284    5.5000
```

Invisibly for the user, Chebfun defines `D2` as a differential operator (not a matrix) from the class `chebop` where boundary conditions are among many of its available characteristics. For the presentation of Chebfun's symbolic computation ability, one may define any function f (as a entity from class `chebfun`) and apply the operator on it. In the example to come, not just complicated function f is defined and differentiated, but also the solution of the backward equation with Dirichlet boundary conditions is found. Note, that results from the previous computations (implicit functions) are used as arguments at the right-hand side of the equation. This is a feature MATLAB itself is not able to offer.

```
d = domain(-5,5);           % define domain
% some wild function
f = chebfun('exp(-x.^2/16).*(1+.2*cos(10*x))',[-5,5]);
plot(f)

D1 = diff(d,1);             % differential operator
g = D1*f;                   % apply the operator to a function
plot(g)

D2 = diff(d,2);             % define another operator
D2.bc = 'dirichlet';        % including boundary conditions
h = D2\(f+g);               % find solution of the equation D2*h = f+g
plot(h)
```

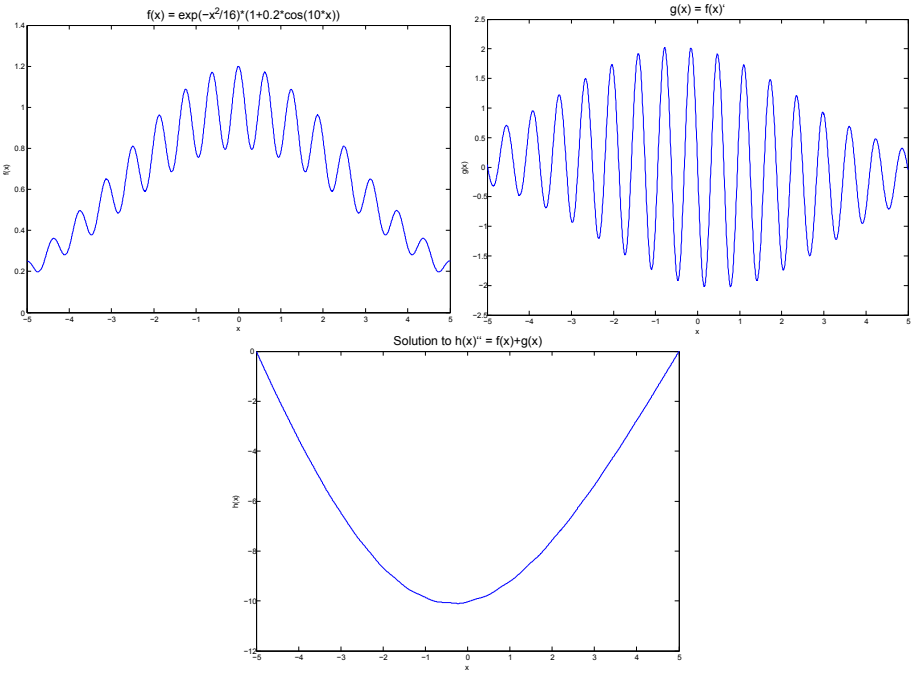


Figure 7: A function defined via symbolic Chebfun, its derivative and the solution to a differential equation with boundary conditions.

Results of the computation are on Figure 7. If we want to learn more about the function `h` obtained, let us just suppress the semicolon in the command line:

```
h
```

```
h =

   chebfun column (1 smooth piece)
       interval       length   endpoint values
[      -5,       5]       88 -6.2e-13 -1.4e-15
vertical scale =  10
```

These are all important information about the function. First, it made us sure, that the Dirichlet boundary conditions are satisfied due to the endpoint values printed out. Moreover, the `length` property informs about the degree of Chebyshev polynomial used.

The last overloaded function we will concern about is `eigs`. The singular matrix `B` on the right-hand side of 27 is automatically generated while recalling the boundary conditions, thus it is not surprising that for obtaining eigenvectors and eigenvalues of an operator with given boundary conditions one only needs to use the command `eigs`.[5]

```
d = domain(-1,1);
D2 = diff(d,2);
D2.bc = 'dirichlet';
format long
eigs(-D2,5)
```

```
ans =

   2.467401100272662
   9.869604401089157
  22.206609902451021
  39.478417604357745
  61.685027506808595
```

One may observe, that the result is accurate up to 12 digits (compare to the exact result (9)).

We will not go further into details regarding the features of Chebfun and redirect an interested reader to http://www2.maths.ox.ac.uk/chebfun/ where one can find not just the guide but also illustrative examples.

[5] In MATLAB notation, `eigs` call for eigenvalues and eigenfunctions of a stiff problem. However, in Chebfun, `eigs` is meant to return a set of finite number of eigenvalues, respectively eigenvectors, regardless of the problem posedness.

In Chebfun, there is a class called `domain` where you can specify the required computational interval. However, Chebfun is not capable of considering other types of grids yet (or rectangle in 2D version even though Chebfun is at heart 1D program). Hence, the goal of the present thesis, among all, is to extend the current features by a new class called `graph`, that provides the user the opportunity to define functions and operators on graph-like domains given there are matching conditions satisfied at the internal vertices. More specifically, we are interested into function computing spectra of quantum graphs as described above.

4.2.2 Developing `graph` class

First of all, the underlying metric graph should be defined. For this purpose, we introduced a new class `graph` where incidence matrix is on the input. Incidence matrix I is defined as $N \times M$ matrix where N is the number of vertices and M is the number of edges such that

$$I_{ij} = \begin{cases} -1 & \text{if } V_i \text{ is the left endpoint of the } j\text{-th edge} \\ 1 & \text{if } V_i \text{ is the right endpoint of the } j\text{-th edge} \\ 0 & \text{otherwise.} \end{cases}$$

Let us start with the simplest case of two connected intervals with edge lengths 1 and 5:

```
A = [-1 1 0;
    0 -1 1]';       % incidence matrix
len = [1 5];        % edge lengths
t = graph(A,len)
```

```
t =

   2x3 graph defined by incidence matrix:
-1  0
 1 -1
 0  1
   with 2 edges and 3 vertices
   and edge lengths: 1  5
```

In the preceding subsection, we demonstrated how Dirichlet or Neumann boundary conditions can be introduced on a single interval. However, in more topologically complicated cases we wish to impose the matching conditions as well. The method is based on the same pattern, but needs one to be more careful with the signs and row/columns position.

A function `grapheig` has been developed in the `graph` directory. In broad terms, providing the incidence matrix, edge lengths and number of eigenvalues requested (and optionally boundary conditions and potentials), the set of eigenvalues and eigenvectors is returned.

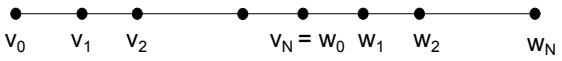


Figure 8: The grid for two connected intervals.

Mathematically, for two intervals glued together (see the grid on Figure 8) standard matching conditions mean that we are solving the discretized problem (following the formalism introduced in (5)):

$$\begin{cases} -D_N^{(2)} v = \lambda v, \\ -D_N^{(2)} w = \lambda w, \\ v_0' = w_N' = 0, \\ v_N = w_0, \\ v_N' = w_0', \end{cases} \tag{28}$$

where $v = [v_0, \dots, v_N]^T$ and $w = [w_0, \dots w_N]^T$. The Laplace matrix of the whole system (before implementing the matching conditions) is now block diagonal matrix of the size $2(N+1) \times 2(N+1)$. We aim to get the linear problem in the generalized form (27).

While calling for `grapheig`, second order differentiation matrix with boundary conditions is created:

```
D2(5)
```

```
D2(5) =

  Columns 1 through 7

   13.1962  -21.3485   12.0000   -6.6515    2.8038         0         0
   -1.0000    4.0000   -6.0000    4.0000   -1.0000         0         0
    2.8038   -6.6515   12.0000  -21.3485   13.1962         0         0
         0         0         0         0         0   13.1962  -21.3485
         0         0         0         0         0   -1.0000    4.0000
         0         0         0         0         0    2.8038   -6.6515
    5.5000   -6.8284    2.0000   -1.1716    0.5000         0         0
         0         0         0         0         0    0.5000   -1.1716
    0.5000   -1.1716    2.0000   -6.8284    5.5000    5.5000   -6.8284
         0         0         0         0    1.0000   -1.0000         0

  Columns 8 through 10

         0         0         0
         0         0         0
         0         0         0
   12.0000   -6.6515    2.8038
   -6.0000    4.0000   -1.0000
```

```
   12.0000  -21.3485   13.1962
         0         0         0
    2.0000   -6.8284    5.5000
    2.0000   -1.1716    0.5000
         0         0         0
```

The matching conditions in (28) indicate that the first and last rows of both $D_N^{(2)}$'s shall be replaced by the respective identity or first order rows. This may be seen from the output: Neumann boundary conditions are implemented in 7th and 8th rows while the continuity conditions are placed in the last two rows. At the same time, singular matrix B has to be adjusted accordingly. In this specific case, B becomes identity matrix of the same size with the last four diagonal elements excluded and replaced by zeros. Note, that we always end up with a square matrix `D2`.

For instance, resuming the string graph `t` defined above, we impose Neumann boundary conditions at the endpoints (if not specified, Neumann is a default choice). The electric potential is set to be zero (for potential usage see Section 4.2.3). The result of the computation is listed below:

```
bcd = [];                   % vertices with Dirichlet BC
bcn = [1,3];                % vertices with Neumann BC
numeig = 3;                 % number of eigenvalues required
q = @(x,u) 0.*x.*u;         % electric potential
[V,lam] = grapheig(t, numeig, bcd, bcn, q)
```

```
V =

   chebfun column 1 (2 smooth pieces)
       interval       length   endpoint values
[       0,       1]       15     0.41     0.41
[       1,       6]       20     0.41     0.41
Total length = 35   vertical scale = 0.41

   chebfun column 2 (2 smooth pieces)
       interval       length   endpoint values
[       0,       1]       32     0.58      0.5
[       1,       6]       25      0.5    -0.58
Total length = 57   vertical scale = 0.58

   chebfun column 3 (2 smooth pieces)
       interval       length   endpoint values
[       0,       1]       28     0.58     0.29
[       1,       6]       29     0.29     0.58
Total length = 57   vertical scale = 0.58
```

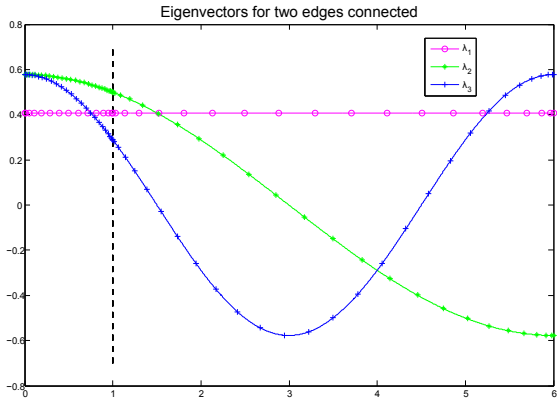


Figure 9: First three eigenvectors for two edges (of lengths 1 and 5) glued into one.

```
lam =

   0.000000000545866
   0.274155678623759
   1.096622711782822
```

A simple plot is showed on Figure 9. The dashed line represents the merge point as the edge lengths are 1 and 5. It is also worth noticing how the Chebyshev points are spread along the lines. They are clustered near the endpoints, thus their density is obviously growing at the breakpoint.

For the results to be more expressive, we define the visualization function `plottree` which is invoked when plotting the results. The incidence matrix and the chebfuns to be plotted are required on the input, optionally also the vertices defined on the plane. If not specified, they are symmetrically spread along the unit circle.

To present the computational power and abilities of `plottree` we visualize the first two eigenvectors of an equilateral graph consisting of 6 vertices and 12 edges:

```
A = [-1 1 0 0 0 0;
     0 -1 1 0 0 0;
     0 0 -1 0 1 0;
     0 0 0 -1 0 1;
     0 0 -1 1 0 0;
     0 0 0 -1 1 0;
     0 0 0 0 -1 1;
     -1 0 0 0 0 1;
     1 0 0 -1 0 0;
     0 -1 0 1 0 0;
```

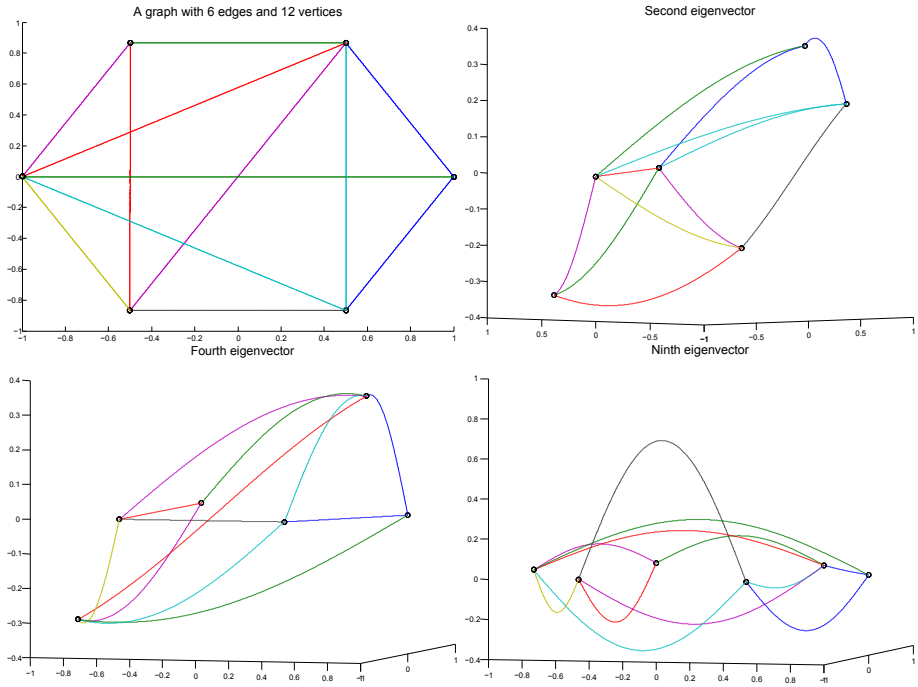


Figure 10: A graph and some of its eigenvectors.

```
    0 -1 0 0 0 1;
    0 -1 0 0 1 0]';      % incidence matrix
len = 2;                 % all lengths set to 2
numeig = 9;              % number of eigenvectors
[V,lam] = grapheig(A, numeig, len);
plottree(V,A)
```

The result is plotted on Figure 10. We picked some of the first eigenvectors while keeping in mind that for connected graph with standard matching conditions the first eigenvector is always a constant function.

However, writing down the incidence matrix and other input arguments gets tedious as the number of edges grows. One also needs to keep track of the node and edge numbering while changing their boundary conditions or lengths respectively. Another option of defining the problem offers a function `drawgraph`. As the name prompts, it enables the user to draw a graph on the plain canvas, connect the vertices with edges and finally select the boundary conditions (Neumann is default in case none is chosen), all just by mouse clicking on the screen. Invisibly, the program resumes the incidence matrix from the plot, computes the edge lengths and marks the boundary vertices with respective conditions. Successively, the first few eigenvectors are plotted on the graph. This procedure may be followed step-by-step on Figure 11 (even though the reader is advised to try this out by himself/herself).

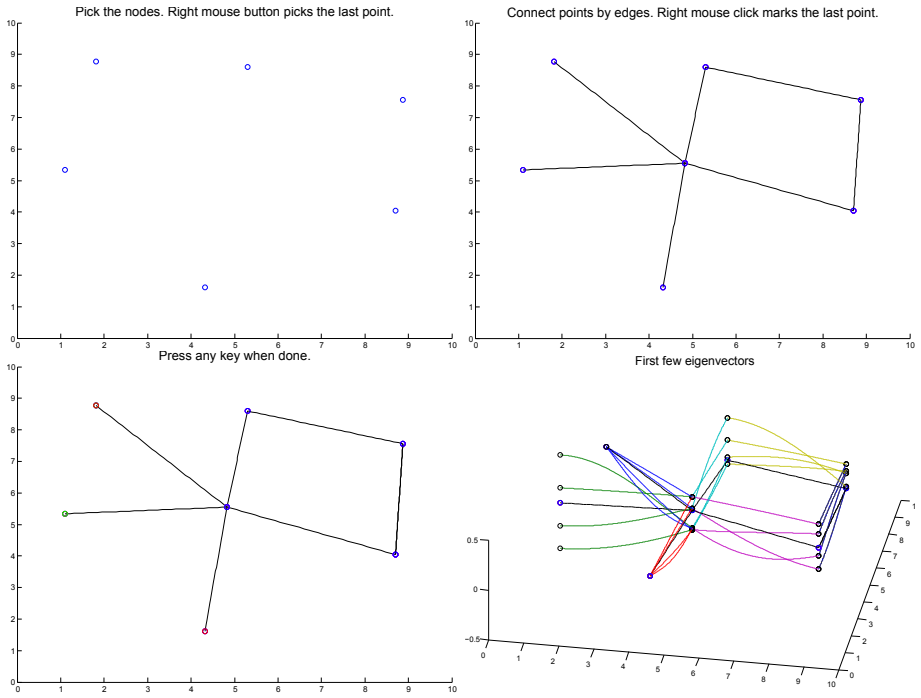


Figure 11: Usage of `drawgraph`: pick the points, connect them with edges, define boundary conditions at pending vertices and wait for Chebfun to do the rest.

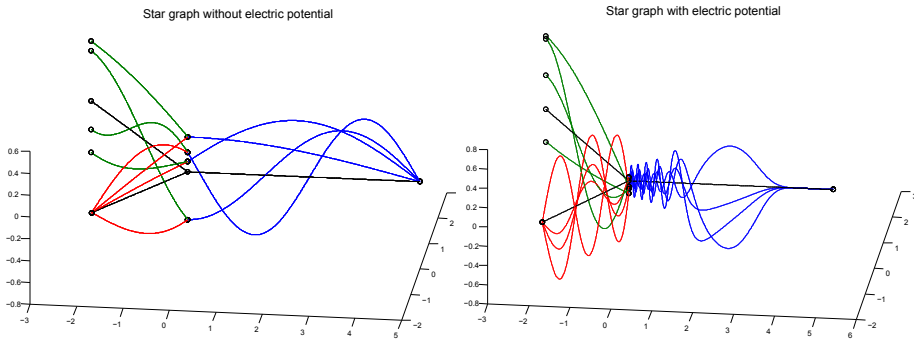


Figure 12: The eigenvectors of a graph affected by a complicated potential lose its trigonometric properties.

4.2.3 Adding potentials

Until now we have considered electric and magnetic potentials to be zero. However, it might come handy to be able to express the Laplacian in more general form 1 as Schrödinger operator. This requires subtle changes in the code.

First, for to obtain Laplacian as in (3) we implement an electric potential q and modify the discretized problem (26) to

$$-D_N^{(2)}u + Qu = \lambda u, \quad u = [u_0, \dots u_N]^T, \tag{29}$$

where Q is $(N+1) \times (N+1)$ diagonal matrix

$$Q = \begin{pmatrix} q_0 & 0 & \dots & 0 \\ 0 & q_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & q_N \end{pmatrix},$$

and q_i is the potential q evaluated in the Chebyshev node x_i (20):

$$q_i := q(x_i).$$

Similarly, for k edges, the potential matrix Q would be diagonal of the size $k(N+1) \times k(N+1)$ with $q_0^1, q_1^1 \dots, q_N^1, q_0^2, \dots q_N^2, \dots, q_N^k$ on the diagonal, where q_i^j is q evaluated the ith node on jth edge. Straightforward computation follows this pattern and allows one to specify the potential using the function `getpotential`. The syntax is as follows:

```
A = [-1 1 0 0;
    0 -1 1 0;
    0 -1 0 1]';                    % incidence matrix
len = [5,3,2];                     % edge lengths
G = graph(A,len);                  % build graph
```

```
numeigs = 4;                              % number of eigenvalues
bcd = [1,4];                              % Dirichlet BC in two vertices
bcn = [3];                                % Neumann BC in one vertex

% potentials are stored in a cell
q{1} = @(x,u) 20*x.^5.*u;                 % potential on the first edge
q{2} = @(x,u) cos(30*x).*u;               % on the second edge
q{3} = @(x,u) 20*exp(-x.^3).*u;           % on the third edge
[V1,D1] = grapheig(G,numeigs,bcd,bcn);      % without potential
[V2,D2] = grapheig(G,numeigs,bcd,bcn,q);    % with potential

a = len(1)*1;                             % define the nodes in space
b = len(2)*(-1+1i)*(sqrt(3)/2);
c = len(3)*(-1-1i)*(sqrt(3)/2);
plottree(V1,A,[a 0 b c])
plottree(V2,A,[a 0 b c])
```

The plots are presented on Figure 12. Here we chose the potentials to be successively: $20x^5, \cos(30x), 20\exp(-x^3)$. One may observe that the complicated potentials affect the eigenvectors in the sense that they are no longer trigonometric.

While defining the electric potentials for computations, keep in mind that each edge is viewed as a subset $[-\ell/2, \ell/2]$ of the real line where ℓ is the respective edge length.

Magnetic potential a in the Schrödinger operator (1) may be partially excluded by introducing an appropriate transformation. Let us define the unitary mapping

$$U_a|_{E_n} : u(x)|_{E_n} \mapsto \exp\left(-i\int_{x_{2n-1}}^{x} a(y)dy\right) u(x)|_{E_n}, \tag{30}$$

which transforms the magnetic Schrödinger operator to

$$L_{q,a} = U_a^{-1} L_{q,0} U_a.$$

However, the magnetic potential still remains present in the boundary terms. Defining $\tilde{u}(x)|_{E_n} = \exp\left(-i\int_{x_{2n-1}}^{x} a(y)dy\right) u(x)|_{E_n}$ we can rewrite the endpoint values as

$$\tilde{u}(x_{2n-1}) = u(x_{2n-1}), \quad \tilde{u}(x_{2n}) = e^{-i\varphi_n} u(x_{2n}), \tag{31}$$

where φ_n denotes $\varphi_n := \int_{x_{2n-1}}^{x_{2n}} a(y)dy$. Similarly, recalling the definition (2), the derivatives obey

$$\begin{aligned} \partial\tilde{u}(x_{2n-1}) &= \partial u(x_{2n-1}), \\ \partial\tilde{u}(x_{2n}) &= e^{-i\varphi_n}\partial u(x_{2n})), \end{aligned} \tag{32}$$

where $\partial u(x_{2n-1})$ and $\partial u(x_{2n})$ denote the normal derivatives defined by (5).

The extension of the discrete problem (29) thus yields

$$-D_N^{(2)}\tilde{u} + Q\tilde{u} = \lambda\tilde{u}, \quad \tilde{u} = [\tilde{u}_0, \dots \tilde{u}_N]^T,$$

where the matching and boundary conditions are making use of the formulation (31) and (32). The algorithm `grapheig` is capable of invoking magnetic potential too, for example, we apply magnetic potential to the graph defined above:

```
q = @(x,u) 0.*x.*u;           % zero electric potential
a = @(x) 3.*exp(x).*x.^3;     % magnetic potential
[V,D] = grapheig(G,numeigs,bcd,bcn,q,a)
```

```
D =

   0.0987 + 0.0000i
   0.3492 - 0.0000i
   0.8883 + 0.0000i
   1.7488 + 0.0000i
```

4.2.4 Implementation

All functions described above plus some supporting procedures have been implemented and included into the `graph` directory in Chebfun. As it is not a part of the official release yet, you may download the package which has been created only for the purpose of the present thesis in http://gemma.ujf.cas.cz/∼malenova/download.html and place it to your MATLAB path. Besides, the code is attached in Appendix.

As the output of `drawgraph` and `grapheig` respectively, we get the eigenvectors and eigenvalues corresponding to the general quantum graph defined by its incidence matrix, edge lengths and optionally boundary conditions or electric and magnetic potentials. The computations are very accurate for spectral methods generate the error dropping rapidly.

5 Applications

Once we have a program computing the spectrum of a magnetic Schrödinger operator on a general graph, it is handy to draw our attention to inverse problems. Does the spectrum carry any information about the properties of the graph? We investigated some interesting issues including the trace formula and the spectral gap.

5.1 Trace formula

The *Euler characteristic* χ for graphs is given by

$$\chi = M - N, \tag{33}$$

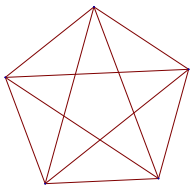


Figure 13: Complete graph with 5 nodes denoted by K_5.

where M, N is number of vertices and edges respectively. Trees have $\chi = 1$, whereas all the other connected graphs have $\chi \leq 0$.

As a function of a quantum graph's spectrum, Euler characteristic χ also comes out from a trace formula [21] as a byproduct:

$$\chi = 2m_s(0) + 2 \lim_{t\to\infty} \sum_{k_n \neq 0} \cos(k_n/t) \left(\frac{\sin(k_n/2t)}{k_n/2t} \right)^2, \tag{34}$$

where $m_s(0)$ is the spectral multiplicity of eigenvalue 0 and k_n^2 is n-th eigenvalue. Limit and sum may not be exchanged for the expression not to diverge (as $\cos(k_n/t) \left(\frac{\sin(k_n/2t)}{k_n/2t} \right)^2 \longrightarrow 1$ providing $t \to \infty$).

Let us define the residual

$$R_N := \sum_{n=N}^{\infty} \cos(k_n/t) \left(\frac{\sin(k_n/2t)}{k_n/2t} \right)^2.$$

A rough estimate is made utilizing the upper bound for goniometric functions and Weyl's asymptotics $k_n \sim \frac{\pi n}{L}$, where L is the total length of the graph:

$$|R_N| = \left| \sum_{n=N}^{\infty} \underbrace{\cos\left(\frac{\pi n}{Lt}\right)}_{\leq 1} \underbrace{\sin^2\left(\frac{\pi n}{2Lt}\right)}_{\leq 1} \left(\frac{2Lt}{\pi n}\right)^2 \right| \leq \frac{4L^2t^2}{\pi^2} \left| \sum_{N}^{\infty} \frac{1}{n^2} \right| \leq$$
$$\leq \frac{4L^2t^2}{\pi^2} \int_{N-1}^{\infty} \frac{1}{x^2}\, dx = \frac{4L^2t^2}{\pi^2} \frac{1}{N-1}. \tag{35}$$

Example 1. Say we want to compute Euler characteristic of a K_5 graph (Figure 13) using the formula (34) achieving precision $|R_N| < 0.25$. The estimate (35) states, that one would need to take approximately $N = 650$ terms at time $t = 1$ to meet the requirements.

As to theoretical estimation, the number of terms necessary for the numerical computation is far lower. According to (33), the correct answer is $\chi = 5 - 10 = -5$. We may take use of Proposition 2.3 and compute only the first period which makes the process much more precise and faster, since we need large amount of eigenvalues. In Figure 14 one may observe the convergence of χ for various

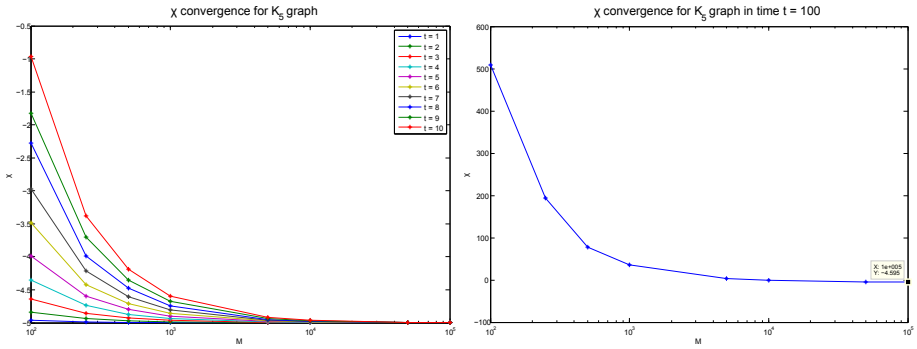


Figure 14: Euler characteristic χ for K_5 graph (Figure (13)) at time $t = 1$. On the x-axis, there is number of terms used.

times. In case $t = 1$ the curve needs barely $N = 20$ terms to achieve the same precision as required.

5.2 Spectral gap

In the literature, the spectral gap has been previously investigated on discrete (combinatorial) graphs and denotes the second eigenvalue of the Laplacian (on a connected graph). It is also referred to as algebraic connectivity or Fiedler value. Recalling the well-known properties for discrete graphs, we investigate the spectral gap on continuous quantum graphs too and compare the results obtained. It is shown that the correspondence is not one-to-one as the behavior differs while adding or deleting the edges which is the main objective of the forthcoming section.

Firstly, let us a have a fixed vertex set. The theory claims that the combinatorial graph's algebraic connectivity is a monotonous function of the set of edges in the sense that adding an edge (without altering the set of vertices) leads to an increase of its connectivity. Section 6 involves proof of this proposition.

However, the same can not be said about the spectral gap of a quantum graph. As a counterexample, we take an equilateral hexagon-like graph with 13 edges (first line of Figure 15). Spectral gap and the corresponding eigenfunction are again computed by calling `grapheig`, for more details see Section 4.2.

Afterwards, two graphs are created from the original one by cutting off an edge, one each. In the first case, the edge deletion causes an *increase* of the spectral gap, while for the latter graph where different edge has been omitted, the spectral gap *decreases* moderately. Indeed, both options are possible irrespectively of the graphs having the same total length, number of vertices and edges. The precise spectral gap values obtained are successively:

```
>> lam(2)                % spectral gap of the...

ans =
```

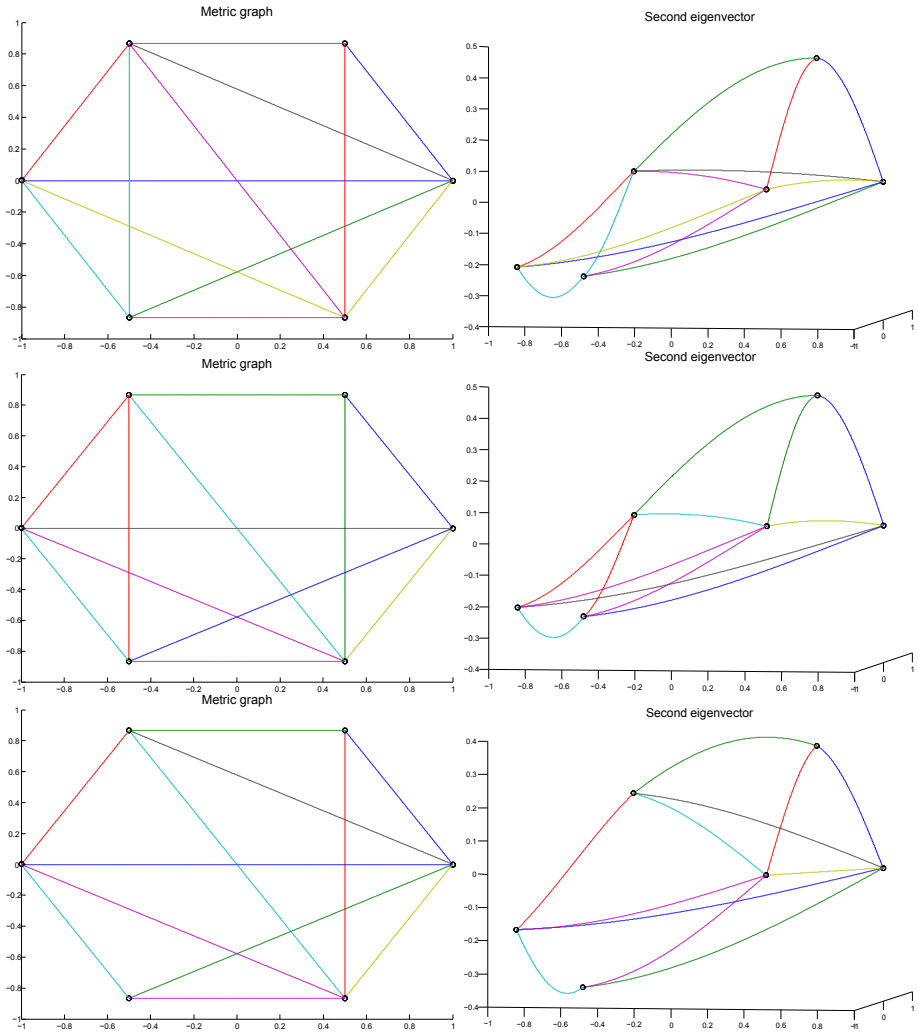


Figure 15: The first graph consists of 13 edges and in the two following graphs, an edge is cut off from each. However, for the second graph the spectral gap *increases* while in the third graph it *decreases*. The figures in the right-hand side column represent the eigenfunctions corresponding to the spectral gap.

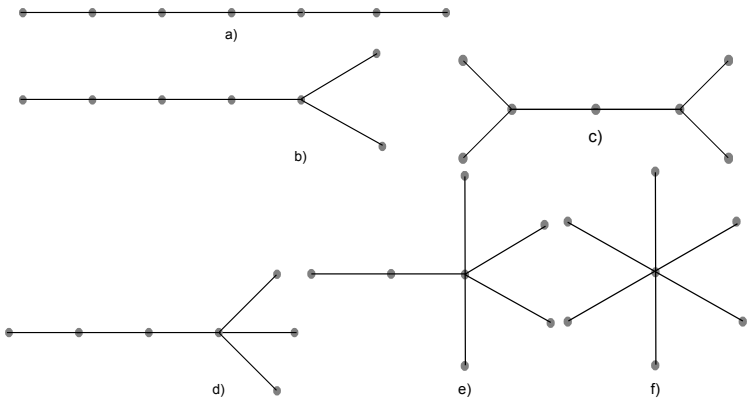


Figure 16: Equilateral graphs of the same total length. One can see that the more branching, the higher is the spectral gap. The spectral gap is successively: a) 0.0685, b) 0.0786 c) 0.0947 d) 0.1086 e) 0.1963 f) 0.6169

```
0.534093660067314    % ... first graphs = original
0.536312125512350    % ... second graph -> increase
0.476145255583854    % ... third graph -> decrease
```

This discovery lead to the investigation on what are the conditions for the spectral gap to decrese/increase and how is this connected to the algebraic connectivity of the underlying metric graph. The answer is partially given in Section 6.

Let us further consider a single string of the length $\mathcal{L}$ (Figure 16a). Spectral gap of the standard Laplacian is well-known from (11): $\lambda_1 = \frac{\pi^2}{\mathcal{L}^2}$. Now, let us modify this graph by parallelizing two of its edges while keeping the same total length $\mathcal{L}$, see Figure 16b. This splitting is sometimes called a *branching*, i. e. including a vertex having valency at least 3. It has been numerically computed that this causes increase of the second eigenvalue. Indeed, the spectral gap raised from 0.0685 to 0.0786, three branches (Figure 16d) brought further increase of 0.1086, given we have equilateral graphs with all edge lengths equal to 2.

Generally, if we had a building kit consisting of 7 nodes and 6 edges, each of the possible combinations would have lower spectral gap than the string graph. The list of options is not complete on Figure 16 (there are three missing), however, the computation suggests a hypothesis, that the simple string of all graphs of the same total length has the lowest spectral gap. Proof of this conjecture is the content of Section 6.3.

Finally, as we just presented, comparing string to any other graph of the same total length always gives higher λ_1 for the latter one. However, one may expect, that enlarging one of the edges of the general graph, there should be a point where the graph begins to behave like a string. Asymptotically, the behavior of the two graphs should be the same.

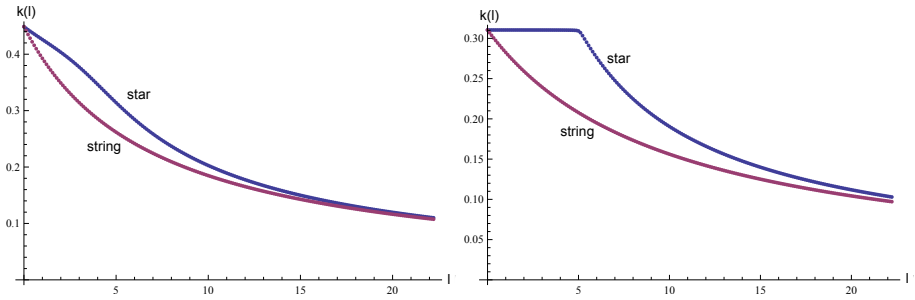


Figure 17: Comparison of the spectral gap for star and string graph. One length l of the star graph is variable, the other two are fixed. Edge lengths are a) $l_1 = 2, l_2 = 5$ and b) $l_1 = l_2 = 5$.

Let us show this behavior on the 3-star graph. As shown above, the string's first excited eigenvalue is $\frac{\pi^2}{L^2}$ while the spectral gap for star graph is given by (16).

In Figure 17a, the comparison is performed. There are two edges of the star graph fixed ($l_1 = 2, l_2 = 5$) and l_3 is variable on the horizontal axis. Blue plot represents the star while red line depicts the string graph of the same total length, i. e. the eigenvalue $\frac{\pi^2}{(l_1+l_2+l_3)^2}$.

One can observe that the difference increases for small l_3. The length of the edges in the star graph are comparable in this regime. However, while increasing one of the lengths to infinity, the star graph begins to act more and more like a string and their spectra almost coincide.

This is even more distinct in the special case of two edges of the star graph having the same length. The case $l_1 = l_3 = l$ was analytically resolved in Section 3.4.

Numerical solution is presented in Figure 17b. We set the values to $l_1 = l_3 = 5$ and adjust the third star graph edge. In the first part, the eigenvalue follows the first type of solution given by (17), what changes to solution of (18) while crossing the point $l = 5$.

The first type of solution corresponds to the case when the eigenfunction on the edge e_2 stays constant. Whereas after crossing the equilateral state $l_1 = l_2 = l_3$ the solution becomes wavy on all the edges.

The spectral asymptotics is particulary considered in Theorem 6.6 of Section 6.2.2 which proves that adding an edge to a graph where the length of this additional edge is greater than the total length of the original graph makes the spectral gap always decrease.

6 Spectral gap

Based on the observations from the previous section we formulate specific theorems and successively prove the statements obtained. For to compare the

behavior, we first begin with similar claims regarding the discrete graphs. In spite they in general do not show the same properties as continuous graphs, some degree of coherence has been observed.

The forthcoming section concerning the spectral gap is the content of the paper [23] that is about to be published.

6.1 Discrete graphs

Let G be a discrete graph with M vertices and N edges connecting some of the vertices. Then the corresponding Laplace operator $L(G)$ is defined on the finite dimensional space $\ell_2(G) = \mathbb{C}^M$ by the following formula

$$(L(G)\psi)(m) = \sum_{n \sim m} (\psi(m) - \psi(n)), \tag{36}$$

where the sum is taken over all neighboring vertices. The Laplace operator can also be defined using the connectivity matrix $C = \{c_{nm}\}$

$$c_{nm} = \begin{cases} 1, & \text{the vertices } n \text{ and } m \text{ are neighbors,} \\ & \textit{i.e.} \text{ connected by an edge;} \\ 0, & \text{otherwise,} \end{cases}$$

and the valence matrix $V = \operatorname{diag}\{v_1, v_2, \ldots, v_M\}$, where v_m are the valencies (degrees) of the corresponding vertices

$$L(G) = V - C,$$

which corresponds to the matrix realization of the operator $L(G)$ in the canonical basis given by the vertices. In the literature one may find other definitions of discrete Laplacians. In [6], [7] one uses essentially the congruent matrix

$$\mathbb{L}(G) = V^{-1/2} L(G) V^{-1/2} = I - V^{-1/2} C V^{-1/2}. \tag{37}$$

Such a definition of the Laplacian matrix is consistent with the eigenvalue analysis in spectral geometry. The following Laplacian matrix connected with the averaging operator is similar to (37):

$$\mathbf{L}(G) = V^{-1} L(G) = V^{-1/2} \mathbb{L}(G) V^{1/2}. \tag{38}$$

The operator (38) is important for studies of quantum graphs, since its eigenvalues are closely related to the spectrum of the corresponding equilateral graphs.

We now discuss briefly spectral properties of the standard (sometimes called combinatorial) Laplace matrix $L(G)$ given by (36), first of all in relation to the set of edges.

Since the Laplace operator is uniquely defined by the discrete graph G its eigenvalues $\lambda_0 \le \lambda_1 \le \cdots \le \lambda_{M-1}$ are usually referred to as the eigenvalues of G. The ground state corresponding to $\lambda_0 = 0$ has eigenfunction $\psi_0 = 1$, where $1 \in \mathbb{C}^M$ denotes the vector built up of ones on G. The multiplicity of

the ground state coincides with the number of connected components in G. In order to avoid artificial complications only connected graphs will be considered in the sequel.

Spectral gap is a monotonous function of the set of edges, in other words, cutting off an edge always causes drop of the second eigenvalue or keeps it unchanged, provided we have the same set of vertices.

Proposition 6.1. *Let G be a connected discrete graph and let G' be a discrete graph obtained from G by adding one edge between the vertices m_1 and m_2. Let L denote the discrete Laplacian defined by* (36)*. Then the following holds:*

1. *The first excited eigenvalues satisfy the inequality:*

$$\lambda_1(G) \leq \lambda_1(G').$$

2. *The equality $\lambda_1(G) = \lambda_1(G')$ holds if and only if the second eigenfunction ψ_1^G on the graph G may be chosen attaining equal values at the vertices m_1 and m_2*

$$\psi_1^G(m_1) = \psi_1^G(m_2).$$

Proof. The first statement follows from the fact that

$$L(G') - L(G) = \begin{pmatrix} & \vdots & & \vdots & \\ \dots & 1 & \dots & -1 & \dots \\ & \vdots & & \vdots & \\ \dots & -1 & \dots & 1 & \dots \\ & \vdots & & \vdots & \end{pmatrix} \tag{39}$$

is a matrix with just four non-zero entries. It is easy to see that the matrix is positive semi-definite, since the eigenvalues are 0 (with the multiplicity $M-1$) and 2 (simple eigenvalue) and therefore $L(G') - L(G) \geq 0$ which implies the first statement.

To prove the last assertion let us recall that $\lambda_1(G')$ can be calculated using Rayleigh quotient

$$\lambda_1(G') = \min_{\psi \perp 1} \frac{\langle \psi, L(G')\psi \rangle}{\langle \psi, \psi \rangle} \geq \min_{\psi \perp 1} \frac{\langle \psi, L(G)\psi \rangle}{\langle \psi, \psi \rangle} = \lambda_1(G).$$

Hence the trial function ψ should be chosen orthogonal to the ground state, *i.e.* having mean value zero. We have equality in the last formula if and only if ψ minimizing the first and the second quotients can be chosen such that $(L(G') - L(G))\psi = 0$, i. e. $\psi(m_1) = \psi(m_2)$.

□

Next we are interested in what happens if we add a pending edge, i. e. an edge connected to the graph at one already existing node.

Proposition 6.2. *Let G be a connected discrete graph and let G' be another graph obtained from G by adding one vertex and one edge between the new vertex and the vertex m_1. Then the following holds:*

1. *The first excited eigenvalues satisfy the following inequality:*

$$\lambda_1(G) \geq \lambda_1(G').$$

2. *The equality $\lambda_1(G) = \lambda_1(G')$ holds if and only if every eigenfunction ψ_1^G corresponding to $\lambda_1(G)$ is equal to zero at m_1*

$$\psi_1^G(m_1) = 0.$$

Proof. Let us define the following vector on G':

$$\varphi(n) := \begin{cases} \psi_1^G(n), & \text{on } G, \\ \psi_1^G(m_1) & \text{on } G' \backslash G. \end{cases}$$

This vector is not orthogonal to the zero energy eigenfunction $1 \in \mathbb{C}^{M+1}$, where we keep the same notation 1 for the vector built up of ones now on G'. Therefore consider the nonzero vector γ shifted by a constant c

$$\gamma(n) := \varphi(n) + c.$$

Here c is chosen so that the orthogonality condition in $l_2(G') = \mathbb{C}^{M+1}$ holds

$$0 = \langle \gamma, 1 \rangle_{l_2(G')} = \underbrace{\langle \psi_1^G, 1 \rangle_{l_2(G)}}_{=0} + \psi_1^G(m_1) + cM',$$

where $M' = M + 1$ is the number of vertices in G'. This implies

$$c = -\frac{\psi_1^G(m_1)}{M'}.$$

Using this vector the following estimate on the first eigenvalue may be obtained:

$$\lambda_1(G') \leq \frac{\langle L(G')\gamma, \gamma \rangle_{l_2(G')}}{\|\gamma\|^2_{l_2(G')}} = \frac{\langle L(G)\psi_1^G, \psi_1^G \rangle_{l_2(G)}}{\|\psi_1^G\|^2_{l_2(G)} + c^2 M + |\psi_1^G(m_1) + c|^2} \leq \lambda_1(G). \quad (40)$$

The last inequality follows from the fact that

$$\langle L(G)\psi_1^G, \psi_1^G \rangle_{l_2(G)} = \lambda_1(G) \|\psi_1^G\|^2_{l_2(G)},$$

and

$$\|\psi_1^G\|^2_{l_2(G)} + c^2 M + |\psi_1^G(m_1) + c|^2 \geq \|\psi_1^G\|^2_{l_2(G)}.$$

Note that we have equality if and only if $c = 0$ and $|\psi_1^G(m_1) + c|^2 = 0$ which implies $\psi_1^G(m_1) = 0$. If there exists a ψ_1^G, such that $\psi_1^G(m_1) \neq 0$, then the inequality in (40) is strict and we get

$$\lambda_1(G) > \lambda_1(G').$$

□

We see that the first excited eigenvalue has a tendency to decrease if a pending edge is attached to a graph. It is clear from the proof that gluing of any connected graph (instead of one edge) would lead to the same result, provided there is just one contact vertex. If the number of contact vertices is larger, then the spectral gap may increase as shown in Proposition 6.1.

Note that a different proof of the first part of Proposition 6.1 may be found in [13], Corollary 3.2. In the same paper, a bit weaker claim related to the first part of Proposition 6.2 is provided as Property 3.3.

6.2 Continuous graphs

In the previous section, the spectral gap of combinatorial graphs has been investigated. We want to explore similar concept for continuous quantum graphs. Our goal here is to study the spectral gap for Laplacians on metric graphs especially in relation to the connectivity of the underlying metric graphs. Successively, we concern about the cases when two vertices are merged into one (Section 6.2.1), an edge is added into the system (Section 6.2.2), an edge is cut at one point (Section 6.2.3) or an edge or its parts are deleted from the original graph (Section 6.2.4). Requiring additional constraints, we may prove an analog of Proposition 6.1 and Proposition 6.2 respectively.

6.2.1 Increasing connectivity - gluing vertices together

Our original idea was to study the behavior of the spectral gap when a new edge is added to the metric graph. But this procedure increases the total length of the graph and it is therefore not surprising that the spectral gap has a tendency to decrease in contrast to Proposition 6.1 (see Theorem 6.5 below). Hence let us start our studies by presenting a direct analog of Proposition 6.1 for quantum graphs. The corresponding theorem answers the following question: what happens to the spectral gap if two vertices in a metric graph are joined into one common vertex. This procedure does not change the set of edges and therefore the total length of the graph is also preserved, but the graph's connectivity increases instead.

Theorem 6.3. *Let Γ be a connected metric graph and let Γ' be another metric graph obtained from Γ by joining together two of its vertices, say V_1 and V_2. Then the following holds:*

1. *The spectral gap satisfies the inequality*

$$\lambda_1(\Gamma) \leq \lambda_1(\Gamma'). \tag{41}$$

2. *The equality $\lambda_1(\Gamma) = \lambda_1(\Gamma')$ holds if and only if the eigenfunction ψ_1 corresponding to the first excited state can be chosen such that*

$$\psi_1(V_1) = \psi_1(V_2). \tag{42}$$

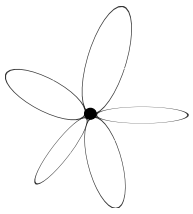


Figure 18: Flower graph.

Proof. The first excited state can be calculated by minimizing the Rayleigh quotient $\frac{\|u'\|^2}{\|u\|^2}$ corresponding to the standard Laplacian over the set of functions from the domain of the quadratic form which are in addition orthogonal to the ground state eigenfunction $\psi_0 = 1$. For the original graph Γ the domain of the quadratic form consists of all $H^1(\Gamma)$ functions which are continuous at all vertices of Γ. The corresponding set for Γ' is characterized by one additional condition $u(V_1) = u(V_2)$ - continuity of the function at the new vertex $V_1 \cup V_2$. Inequality (41) for the corresponding minima follows.

To prove the second statement we first note that if the minimizing function ψ_1 for Γ satisfies in addition (42), then the same function is a minimizer for Γ' and the corresponding eigenvalues coincide. It is clear since the domain of the quadratic form keeps only the continuity of functions at the vertices. Conversely if $\lambda_1(\Gamma) = \lambda_1(\Gamma')$, then the eigenfunction for $L^{\mathrm{st}}(\Gamma')$ is also a minimizer for the Rayleigh quotient for Γ and therefore is an eigenfunction for $L^{\mathrm{st}}(\Gamma)$ satisfying in addition (42).

□

Proposition 6.1 and Theorem 6.3 appear to be rather similar at first glance. But the reasons for the spectral gap to increase are different. In the case of discrete graphs the difference between the Laplace operators is a nonnegative matrix. For quantum graphs the differential operators are identical, but inequality (41) is valid due to the fact that the opposite inequality holds for the domains of the quadratic forms.

Corollary 1. *Theorem 6.3 implies that the flower graph consisting of N loops attached to one vertex (Figure 18) has the largest spectral gap among all graphs formed by a given set of edges.*

6.2.2 Adding an edge

Our goal in this section is to study the behavior of the spectral gap of the standard Laplace operator (Definition 3) as an extra edge is added to the metric graph. We start by proving a direct analog of Proposition 6.2.

Theorem 6.4. *Let Γ be a connected metric graph and let Γ' be another graph obtained from Γ by adding one vertex and one edge connecting the new vertex with the vertex V_1.*

1. *The first eigenvalues satisfy the following inequality:*

$$\lambda_1(\Gamma) \geq \lambda_1(\Gamma').$$

2. *The equality $\lambda_1(\Gamma) = \lambda_1(\Gamma')$ holds if and only if every eigenfunction ψ_1 corresponding to $\lambda_1(\Gamma)$ is equal to zero at V_1*

$$\psi_1(V_1) = 0.$$

Proof. Let us define the following function on Γ':

$$f(x) := \begin{cases} \psi_1(x), & x \in \Gamma, \\ \psi_1(V_1) & x \in \Gamma' \backslash \Gamma. \end{cases}$$

This function is in general not orthogonal to the zero energy eigenfunction $1 \in L_2(\Gamma')$. Therefore consider the nonzero function g differed from f by a constant

$$g(x) := f(x) + c,$$

where c is chosen so that the orthogonality condition in $L_2(\Gamma')$ holds

$$0 = \langle g(x), 1 \rangle_{L_2(\Gamma')} = \underbrace{\langle \psi_1, 1 \rangle_{L_2(\Gamma)}}_{=0} + \psi_1(V_1)\ell + c\mathcal{L}',$$

where ℓ and $\mathcal{L}'$ are the length of the added edge and the total length of Γ' respectively. This implies

$$c = -\frac{\psi_1(V_1)\ell}{\mathcal{L}'}.$$

Using this vector the following estimate for the first eigenvalue may be obtained:

$$\lambda_1(\Gamma') \leq \frac{\langle L^{\mathrm{st}} g, g \rangle_{L_2(\Gamma')}}{\|g\|^2_{L_2(\Gamma')}} = \frac{\langle L^{\mathrm{st}} \psi_1, \psi_1 \rangle_{L_2(\Gamma)}}{\|\psi_1\|^2_{L_2(\Gamma)} + c^2\mathcal{L} + |\psi_1(V_1) + c|^2 \ell} \leq \lambda_1(\Gamma).$$

Here $\mathcal{L}$ denotes the total length of the metric graph Γ. The last inequality follows from the fact that

$$\langle L^{\mathrm{st}} \psi_1, \psi_1 \rangle_{L_2(\Gamma)} = \lambda_1(\Gamma) \|\psi_1\|^2,$$

and

$$\|\psi_1\|^2_{L_2(\Gamma)} + c^2\mathcal{L} + |\psi_1(V_1) + c|^2 \ell \geq \|\psi_1\|^2_{L_2(\Gamma)}.$$

Note that in the last expression the equality holds if and only if $c = 0$ and $|\psi_1(V_1) + c|^2 = 0$ which implies $\psi_1(V_1) = 0$. This proves the second assertion. □

Remark. In the proof of the last theorem we did not really use the fact that $\Gamma' \setminus \Gamma$ is an edge. It is straightforward to generalize the theorem for the case where $\Gamma' \setminus \Gamma$ is an arbitrary finite connected graph joined to Γ at one vertex only.

We return now to our original goal and investigate the behavior of the spectral gap when an edge between two vertices is added to a metric graph.

Theorem 6.5. *Let Γ be a connected metric graph and L^{st} the corresponding free Laplace operator. Let Γ' be a metric graph obtained from Γ by adding an edge between the vertices V_1 and V_2. Assume that the eigenfunction ψ_1 corresponding to the first excited eigenvalue can be chosen such that*

$$\psi_1(V_1) = \psi_1(V_2). \tag{43}$$

Then the following inequality for the second eigenvalues holds:

$$\lambda_1(\Gamma) \geq \lambda_1(\Gamma').$$

Proof. To prove the inequality let us consider the eigenfunction $\psi_1(\Gamma)$ for $L^{\mathrm{st}}(\Gamma)$. We introduce a new function on Γ'

$$f(x) = \begin{cases} \psi_1(x), & x \in \Gamma, \\ \psi_1(V_1) \ \ (= \psi_1(V_2)) & x \in \Gamma' \backslash \Gamma. \end{cases}$$

This function is not orthogonal to the constant function. Let us adjust the constant c so that the nonzero function $g(x) = f(x) + c$ is orthogonal to 1 in $L_2(\Gamma')$:[6]

$$0 = \langle g(x), 1 \rangle_{L_2(\Gamma')} = \underbrace{\langle \psi_1(x), 1 \rangle_{L_2(\Gamma)}}_{=0} + \psi_1(V_1)\ell + c\mathcal{L}' = 0,$$

where ℓ is the length of the added edge and $\mathcal{L}'$ is the total length of the graph Γ', as before. We have used that the eigenfunction ψ_1 has mean value zero, *i.e.* is orthogonal to the ground state. This implies

$$c = -\frac{\psi_1(V_1)\ell}{\mathcal{L}'}.$$

Now we are ready to get an estimate for $\lambda_1(\Gamma')$ using Rayleigh quotient

$$\lambda_1(\Gamma') \leq \frac{\langle L^{\mathrm{st}}(\Gamma')g, g \rangle_{L_2(\Gamma')}}{\|g\|^2_{L_2(\Gamma')}}.$$

The numerator and denominator can be evaluated as follows

$$\langle L^{\mathrm{st}}(\Gamma')g, g \rangle_{L_2(\Gamma')} = \langle L^{\mathrm{st}}(\Gamma)\psi_1, \psi_1 \rangle_{L_2(\Gamma)} = \lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)},$$

$$\begin{aligned} \|g\|^2_{L_2(\Gamma')} &= \|\psi_1 + c\|^2_{L_2(\Gamma)} + |\psi_1(V_1) + c|^2\ell = \\ &= \|\psi_1\|^2_{L_2(\Gamma)} + c^2\mathcal{L} + |\psi_1(V_1) + c|^2\ell \geq \\ &\geq \|\psi_1\|^2_{L_2(\Gamma)} \end{aligned}$$

This implies the inequality

$$\lambda_1(\Gamma) \geq \lambda_1(\Gamma'),$$

that was to be proven. □

[6] In what follows we are going to use the same notation 1 for the functions identically equal to one on both metric graphs Γ and Γ'.

Let us illustrate the above theorem by a couple of examples:

Example 2. Let Γ be the graph formed of one edge of length a. The spectrum of $L^{\mathrm{st}}(\Gamma)$ is

$$\sigma(L^{\mathrm{st}}(\Gamma)) = \left\{ \left(\frac{\pi}{a}\right)^2 n^2 \right\}_{n=0}^{\infty}.$$

All eigenvalues have multiplicity one.

Consider the graph Γ' obtained from Γ by adding an edge of length b, so that Γ' is formed by two intervals of lengths a and b connected in parallel. The graph Γ' is equivalent to the circle of length $a+b$. The spectrum is:

$$\sigma(L^{\mathrm{st}}(\Gamma')) = \left\{ \left(\frac{2\pi}{a+b}\right)^2 n^2 \right\}_{n=0}^{\infty},$$

where all the eigenvalues except for the ground state have double multiplicity.

Let us study the relation between the first eigenvalues:

$$\lambda_1(\Gamma) = \frac{\pi^2}{a^2}, \;\; \lambda_1(\Gamma') = \frac{4\pi^2}{(a+b)^2}.$$

Any relation between these values is possible:

$$b > a \;\Rightarrow\; \lambda_1(\Gamma) > \lambda_1(\Gamma'),$$
$$b < a \;\Rightarrow\; \lambda_1(\Gamma) < \lambda_1(\Gamma').$$

Therefore the first eigenvalue is not in general a monotonously decreasing function of the set of edges. The spectral gap decreases only if certain additional conditions are satisfied.

Example 3. Consider, in addition to graph Γ' discussed in Example 2 , the graph Γ'' obtained from Γ' by adding another one edge of length c between the same two vertices. Hence Γ'' is formed by three parallel edges of lengths a, b and c. The first eigenfunction for $L^{\mathrm{st}}(\Gamma')$ can always be chosen so that its values at the vertices are equal. Then, in accordance with Theorem 6.5, the first eigenvalue for Γ'' is less or equal to the first eigenvalue for Γ':

$$\lambda_1(\Gamma'') \le \lambda_1(\Gamma').$$

This fact can easily be supported by explicit calculations.

The above examples and proved theorems show that the spectral gap has a tendency to decrease, when a new sufficiently long edge is added. It is not surprising, since addition of an edge increases the total length of the graph, but the eigenvalues satisfy Weyl's law and therefore are asymptotically close to $(\pi n)^2/\mathcal{L}^2$. This behavior is in contrast to the one of discrete graphs provided the set of vertices is not altered.

Condition (43) in Theorem 6.5 is not easy to check for non-trivial graphs and therefore it might be interesting to obtain another explicit sufficient conditions.

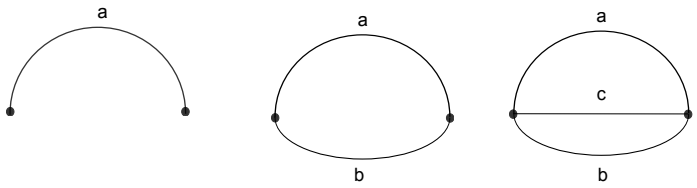


Figure 19: Graphs Γ, Γ', and Γ''.

In what follows we would like to discuss one such geometric condition ensuring that the spectral gap drops as a new edge is added to a graph. The main idea is to compare the length ℓ of the new edge with the total length of the original graph $\mathcal{L}(\Gamma)$. It turns out that if $\ell > \mathcal{L}(\Gamma)$, then the spectral gap always decreases. We have already observed this phenomenon when discussing Example 2, where behavior of λ_1 depended on the ratio between the lengths a and b. If $b \equiv \ell > a \equiv \mathcal{L}(\Gamma)$, then the gap decreases. It is surprising that the same explicit condition holds for arbitrary connected graphs Γ.

Theorem 6.6. *Let Γ be a connected finite compact metric graph of length $\mathcal{L}(\Gamma)$ and let Γ' be a graph constructed from Γ by adding an edge of length ℓ between certain two vertices. If*

$$\ell > \mathcal{L}(\Gamma), \tag{44}$$

then the eigenvalues of the corresponding free Laplacians satisfy the estimate

$$\lambda_1(\Gamma) \geq \lambda_1(\Gamma'). \tag{45}$$

Proof. Let ψ_1 be any eigenfunction corresponding to the first excited eigenvalue $\lambda_1(\Gamma)$ of $L^{\mathrm{st}}(\Gamma)$. It follows that the minimum of the Rayleigh quotient is attained at ψ_1:

$$\lambda_1(\Gamma) = \min_{u \in H^1(\Gamma): u \perp 1} \frac{\|u'\|^2_{L_2(\Gamma)}}{\|u\|^2_{L_2(\Gamma)}} = \frac{\|\psi_1'\|^2_{L_2(\Gamma)}}{\|\psi_1\|^2_{L_2(\Gamma)}},$$

where $H^1(\Gamma)$ denotes the set of H^1-functions on graph Γ. Let us denote by V_1 and V_2 the vertices in Γ, where the new edge E of length ℓ is attached.

The eigenvalue $\lambda_1(\Gamma')$ can again be estimated using Rayleigh quotient

$$\lambda_1(\Gamma') = \min_{u \in H^1(\Gamma'): u \perp 1} \frac{\|u'\|^2_{L_2(\Gamma')}}{\|u\|^2_{L_2(\Gamma')}} \leq \frac{\|g'\|^2_{L_2(\Gamma)}}{\|g\|^2_{L_2(\Gamma)}}, \tag{46}$$

where $g(x)$ is any function in $H^1(\Gamma')$ orthogonal to constant function in $L_2(\Gamma')$. Let us choose the trial function g of the form $g(x) = f(x) + c$ where

$$f(x) := \begin{cases} \psi_1(x), & x \in \Gamma, \\ \gamma_1 + \gamma_2 \sin\left(\frac{\pi x}{\ell}\right) & x \in \Gamma' \backslash \Gamma = E = [-\ell/2, \ell/2], \end{cases} \tag{47}$$

with $\gamma_1 = (\psi_1(V_1) + \psi_1(V_2))/2$ and $\gamma_2 = (\psi_1(V_2) - \psi_1(V_1))/2$. Here we assumed that left end point of the interval is connected to V_1 and the right end point -

to V_2. The function f obviously belongs to $H^1(\Gamma')$, since it is continuous at V_1 and V_2 , but it is not necessarily orthogonal to the ground state eigenfunction 1. The constant c is adjusted in order to ensure the orthogonality

$$\langle g, 1\rangle_{L_2(\Gamma')} = 0$$

holds. The constant c can easily be calculated

$$0 = \langle g, 1\rangle_{L_2(\Gamma')} = c\mathcal{L}' + \underbrace{\langle \psi_1, 1\rangle_{L_2(\Gamma)}}_{=0} + \int_{-\ell/2}^{\ell/2} \left(\gamma_1 + \gamma_2 \sin\left(\frac{\pi x}{\ell}\right)\right) dx = c\mathcal{L}' + \gamma_1 \ell$$

$$\Rightarrow c = -\frac{\gamma_1 \ell}{\mathcal{L}'}. \tag{48}$$

The function g can be used as a trial function in (73) to estimate the spectral gap. Let us begin with computing the denominator using the fact that g is orthogonal to 1

$$\begin{aligned}
\|g\|^2_{L_2(\Gamma')} &= \|f + c\|^2_{L_2(\Gamma')} = \|f\|^2_{L_2(\Gamma')} - \|c\|^2_{L_2(\Gamma')} \\
&= \|\psi_1\|^2_{L_2(\Gamma)} + \int_{-\ell/2}^{\ell/2} \left(\gamma_1 + \gamma_2 \sin\left(\frac{\pi x}{\ell}\right)\right)^2 dx - c^2\mathcal{L}' \\
&= \|\psi_1\|^2_{L_2(\Gamma)} + \ell\gamma_1^2 + \frac{\ell}{2}\gamma_2^2 - c^2\mathcal{L}'
\end{aligned} \tag{49}$$

The numerator yields

$$\begin{aligned}
\|g'\|^2_{L_2(\Gamma')} = \|f'\|^2_{L_2(\Gamma')} &= \|\psi_1'\|^2_{L_2(\Gamma)} + \int_{-\ell/2}^{\ell/2} \left(\gamma_2^2 \frac{\pi^2}{\ell^2} \cos^2\left(\frac{\pi x}{\ell}\right)\right) dx \\
&= \lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} + \gamma_2^2 \frac{\pi^2}{2\ell}
\end{aligned} \tag{50}$$

After plugging (49) and (50) into (73) we obtain

$$\lambda_1(\Gamma') \le \frac{\lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} + \gamma_2^2 \frac{\pi^2}{2\ell}}{\|\psi_1\|^2_{L_2(\Gamma)} + \ell\gamma_1^2 + \frac{\ell}{2}\gamma_2^2 - c^2\mathcal{L}'}.$$

Using (48) the last estimate can be written as

$$\lambda_1(\Gamma') \le \frac{\lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} + \gamma_2^2 \frac{\pi^2}{2\ell}}{\|\psi_1\|^2_{L_2(\Gamma)} + \ell\gamma_1^2\left(1 - \frac{\ell}{\mathcal{L}'}\right) + \frac{\ell}{2}\gamma_2^2} \le \frac{\lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} + \gamma_2^2 \frac{\pi^2}{2\ell}}{\|\psi_1\|^2_{L_2(\Gamma)} + \frac{\ell}{2}\gamma_2^2}, \tag{51}$$

where we used that $\ell < \mathcal{L}' = \mathcal{L} + \ell$. It remains to take into account the following estimate for λ_1 proven in [14, 24]

$$\lambda_1(\Gamma) \ge \left(\frac{\pi}{\mathcal{L}}\right)^2. \tag{52}$$

Then taking into account (44) estimate (51) can be written as

$$\lambda_1(\Gamma') \leq \frac{\lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} + \lambda_1(\Gamma)\gamma_2^2\ell/2}{\|\psi_1\|^2_{L_2(\Gamma)} + \gamma_2^2\ell/2} = \lambda_1(\Gamma). \tag{53}$$

The theorem is proven. □

Estimate (52) was crucial for our proof. It relates the spectral gap to the total length of the metric graph, *i.e.* compares the geometric and spectral properties of quantum graphs. Proposition 6.1 states that the spectral gap increases if an edge is added to the discrete graph. Adding a long edge should correspond to adding a chain of edges to a discrete graph.

The previous theorem gives us a sufficient geometric condition for the spectral gap to decrease. Let us study now the case where the spectral gap is increasing. Similarly, as we proved that adding one edge that is *long enough* always makes the spectral gap smaller (Theorem 6.6), we claim that an edge that is *short enough* makes it grow. We have already seen in Theorem 6.3 that adding an edge of zero length (joining two vertices into one) may lead to an increase of the spectral gap. It turns out that the criterium for a gap to decrease can be formulated explicitly in terms of the eigenfunction on the larger graph. Therefore let us change our point of view and study the behavior of the spectral gap as an edge is deleted.

6.2.3 Decreasing connectivity - cutting edges

In the following section we are going to study the spectral gap's behavior when one of the edges is deleted. The result of such procedure is not obvious, since cutting of an edge decreases the total length of the metric graph and one expects that the first excited eigenvalue increases. On the other hand cutting off an edge decreases the graph's connectivity and therefore the spectral gap is expected to decrease. It is easy to construct examples when one of these two tendencies prevails: Example 2 shows that the spectral gap may both decrease and increase when an edge is deleted.

Let us discuss first what happens when one of the edges is cut in a certain internal point. Let Γ^* be a connected metric graph obtained from a metric graph Γ by cutting one of the edges, say $E_1 = [x_1, x_2]$ at a point $x^* \in (x_1, x_2)$. It will be convenient to denote by x_1^* and x_2^* the points on the two sides of the cut. In other words, the graph Γ^* has precisely the same set of edges and vertices as Γ except that the edge $[x_1, x_2]$ is substituted by two edges $[x_1, x_1^*]$ and $[x_2^*, x_2]$ and two new vertices $V_1^* = \{x_1^*\}$ and $V_2^* = \{x_2^*\}$ are added to the set of vertices.

The spectral gap for the graphs Γ and Γ^* can be calculated by minimizing the same Rayleigh quotient over the set of H^1-functions with zero average. The only difference is that the functions used to calculate $\lambda_1(\Gamma)$ are necessarily continuous at x^*

$$u(x_1^*) = u(x_2^*)$$

(as functions from $H^1[x_1, x_2]$). The functions used in calculating $\lambda_1(\Gamma^*)$ do not necessarily attain the same values at the points x_1^* and x_2^*. It follows that $\lambda_1(\Gamma^*) \leq \lambda_1(\Gamma)$, since the set of admissible functions is larger for Γ^*. If the minimizing function for Γ^* has the same values at x_1^* and x_2^*, then it is also an eigenfunction for $L^{\mathrm{st}}(\Gamma)$ and therefore $\lambda_1(\Gamma^*) = \lambda_1(\Gamma)$. Moreover, if the spectral gap for the graphs is the same, then every function minimizing the quotient for Γ minimizes the quotient for Γ^* as well and therefore satisfies the Neumann condition at x^*. It follows that every eigenfunction for $L^{\mathrm{st}}(\Gamma)$ corresponding to λ_1 is also an eigenfunction for $L^{\mathrm{st}}(\Gamma^*)$. The following theorem is proven.

Theorem 6.7. *Let Γ be a connected metric graph and let Γ^* be another graph obtained from Γ by cutting one of the edges at an internal point x^* producing two new vertices V_1^* and V_2^*.*

1. *The first excited eigenvalues satisfy the following inequality*

$$\lambda_1(\Gamma^*) \leq \lambda_1(\Gamma). \tag{54}$$

2. *If $\lambda_1(\Gamma^*) = \lambda_1(\Gamma)$ then every eigenfunction of $L^{\mathrm{st}}(\Gamma)$ corresponding to $\lambda_1(\Gamma)$ satisfies Neumann condition at the cut point x^*: $\psi_1'(x^*) = 0$. If at least one of the eigenfunctions on Γ^* satisfies $\psi_1^*(V_1^*) = \psi_1^*(V_2^*)$, then $\lambda_1(\Gamma^*) = \lambda_1(\Gamma)$.*

This theorem is a certain reformulation of Theorem 6.3 and implies that the spectral gap has a tendency to decrease when an edge is cut in an internal point. Note that the total length of the graph is preserved this time.

6.2.4 Deleting an edge

Let us study now what happens if an edge is deleted, or if a whole interval is cut away from an edge (without gluing the remaining intervals together). Let Γ be a connected metric graph as before and let Γ^* be a graph obtained from Γ by deleting one of the edges.

The following theorem proves a sufficient condition that guarantees that the spectral gap is decreasing as one of the edges is deleted.

Theorem 6.8. *Let Γ be a connected finite compact metric graph of the total length $\mathcal{L}$ and let Γ^* be another connected metric graph obtained from Γ by deleting one edge of length ℓ between certain vertices V_1 and V_2. Assume in addition that*

$$\left(\max_{\psi_1 : L^{\mathrm{st}}(\Gamma)\psi_1 = \lambda_1\psi_1} \frac{(\psi_1(V_1) - \psi_1(V_2))^2}{(\psi_1(V_1) + \psi_1(V_2))^2} \cot^2 \frac{k_1\ell}{2} - 1 \right) \frac{k_1}{2} \cot \frac{k_1\ell}{2} \geq (\mathcal{L} - \ell)^{-1}, \tag{55}$$

where $\lambda_1(\Gamma) = k_1^2$, $k_1 > 0$ is the first excited eigenvalue of $L^{\mathrm{st}}(\Gamma)$, then

$$\lambda_1(\Gamma^*) \leq \lambda_1(\Gamma). \tag{56}$$

Proof. It will be convenient to denote the edge to be deleted by $E = \Gamma \setminus \Gamma^*$ as well as to introduce notation $\mathcal{L}^* = \mathcal{L} - \ell$ for the total length of Γ^*.

Let us consider any real eigenfunction ψ_1 on Γ corresponding to the eigenvalue $\lambda_1(\Gamma)$. We then define the function $g \in H^1(\Gamma^*)$ by

$$g = \psi_1|_{\Gamma^*} + c,$$

where the constant c is to be adjusted so that g has mean value zero on Γ^*:

$$\langle g, 1\rangle_{L_2(\Gamma^*)} = 0. \tag{57}$$

Straightforward calculations lead to

$$0 = \langle \psi_1, 1\rangle_{L_2(\Gamma^*)} + c\mathcal{L}^* = -\langle \psi_1, 1\rangle_{L_2(E)} + c\mathcal{L}^*$$

$$\Rightarrow c = \frac{\int_E \psi_1(x)dx}{\mathcal{L}^*}. \tag{58}$$

The function g can then be used to estimate the second eigenvalue $\lambda_1(\Gamma^*)$:

$$\lambda_1(\Gamma^*) = \min_{u\in H^1(\Gamma^*): u\perp 1} \frac{\|u'\|^2_{L_2(\Gamma^*)}}{\|u\|^2_{L_2(\Gamma^*)}} \le \frac{\|g'\|^2_{L_2(\Gamma^*)}}{\|g\|^2_{L_2(\Gamma^*)}}. \tag{59}$$

Bearing in mind that $\langle \psi_1, 1\rangle_{L_2(\Gamma)} = 0$ and using (58) we evaluate the denominator in (59) first:

$$\|g\|^2_{L_2(\Gamma^*)} = \|\psi_1 + c\|^2_{L_2(\Gamma^*)} = \int_\Gamma (\psi_1 + c)^2\, dx - \int_E (\psi_1 + c)^2\, dx =$$

$$= \|\psi_1\|^2_{L_2(\Gamma)} - \int_E {\psi_1}^2\, dx - \frac{1}{\mathcal{L}^*}\left(\int_E \psi_1\, dx\right)^2. \tag{60}$$

The numerator similarly yields

$$\|g'\|^2_{L_2(\Gamma^*)} = \int_\Gamma (\psi_1')^2\, dx - \int_E (\psi_1')^2\, dx = \lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} - \int_E (\psi_1')^2\, dx. \tag{61}$$

Plugging (60) and (61) into (59) we arrive at

$$\lambda_1(\Gamma^*) \le \frac{\lambda_1(\Gamma)\|\psi_1\|^2_{L_2(\Gamma)} - \int_E (\psi_1')^2\, dx}{\|\psi_1\|^2_{L_2(\Gamma)} - \int_E {\psi_1}^2\, dx - \frac{1}{\mathcal{L}^*}\left(\int_E \psi_1\, dx\right)^2}. \tag{62}$$

Let us evaluate the integrals appearing in (62) taking into account that ψ_1 is a solution to the Helmholtz equation on the edge E which can be parameterized as $E = [-\ell/2, \ell/2]$ so that $x = -\ell/2$ belongs to V_1 and $x = \ell/2$ to V_2

$$\psi_1|_E(x) = \alpha \sin(k_1 x) + \beta \cos(k_1 x), \tag{63}$$

where

$$\alpha = -\frac{\psi_1(V_1) - \psi_1(V_2)}{2\sin(k_1\ell/2)}, \quad \beta = \frac{\psi_1(V_1) + \psi_1(V_2)}{2\cos(k_1\ell/2)}. \tag{64}$$

Direct calculations imply

$$\begin{aligned}\int_E \psi_1(x)dx &= \frac{2\beta}{k_1}\sin\left(\frac{k_1\ell}{2}\right);\\ \int_E (\psi_1(x))^2dx &= \frac{\alpha^2+\beta^2}{2}\ell - \frac{\alpha^2-\beta^2}{2}\frac{\sin(k_1\ell)}{k_1};\\ \int_E (\psi_1'(x))^2dx &= k_1^2\left(\frac{\alpha^2+\beta^2}{2}\ell + \frac{\alpha^2-\beta^2}{2}\frac{\sin(k_1\ell)}{k_1}\right).\end{aligned}$$

Inserting calculated values into (62) we get

$$\lambda_1(\Gamma^*) \le \lambda_1(\Gamma)\frac{\| \psi_1 \|^2_{L_2(\Gamma)} - \frac{\alpha^2+\beta^2}{2}\ell - \frac{\alpha^2-\beta^2}{2}\frac{\sin(k_1\ell)}{k_1}}{\| \psi_1 \|^2_{L_2(\Gamma)} - \frac{\alpha^2+\beta^2}{2}\ell + \frac{\alpha^2-\beta^2}{2}\frac{\sin(k_1\ell)}{k_1} - \frac{1}{\mathcal{L}^*}\frac{4\beta^2}{\lambda_1(\Gamma)}\sin^2\left(\frac{k_1\ell}{2}\right)}. \tag{65}$$

To guarantee that the quotient is not greater than 1 and therefore $\lambda_1(\Gamma^*) \le \lambda_1(\Gamma)$ it is enough that

$$\begin{aligned}&\frac{\alpha^2-\beta^2}{2}\frac{\sin(k_1\ell)}{k_1} \ge -\frac{\alpha^2-\beta^2}{2}\frac{\sin(k_1\ell)}{k_1} + \frac{1}{\mathcal{L}^*}\frac{4\beta^2}{\lambda_1(\Gamma)}\sin^2\left(\frac{k_1\ell}{2}\right)\\ &\iff \frac{k_1}{2}\left(\frac{\alpha^2}{\beta^2}-1\right)\cot\left(\frac{k_1\ell}{2}\right) \ge (\mathcal{L}^*)^{-1}.\end{aligned} \tag{66}$$

Using (64) the last inequality can be written as

$$\left(\frac{(\psi_1(V_1)-\psi_1(V_2))^2}{(\psi_1(V_1)+\psi_1(V_2))^2}\cot^2\left(\frac{k_1\ell}{2}\right) - 1\right)\frac{k_1}{2}\cot\left(\frac{k_1\ell}{2}\right) \ge (\mathcal{L}^*)^{-1}.$$

Remembering that the eigenfunction ψ_1 could be chosen arbitrary we arrive at (55). □

Roughly speaking, the condition (55) means that the length ℓ is sufficiently small. Indeed, for small ℓ the cotangent term is of order $1/\ell$. Therefore the right-hand side of (55) is of order $1/\ell^3$ and thus growing to infinity as ℓ decreases.

Let us apply the above theorem to obtain an estimate for the length of the piece that can be cut from an edge so that the spectral gap still decreases. It turns out that such an estimate can be given in terms of the eigenfunction ψ_1 corresponding to the first excited eigenvalue. Consider any edge of Γ say $E_1 = [x_1, x_2]$ and choose an arbitrary internal point $x^* \in (x_1, x_2)$. Assume that we cut away an interval of length ℓ centered at x^*. Of course the length ℓ should satisfy the obvious geometric condition: $x_1 \le x^* - \ell/2$ and $x^* + \ell/2 \le x_2$. We assume in addition that

$$\ell < \frac{\pi}{2k_1} \tag{67}$$

guaranteeing in particular that the cotangent function in (55) is positive.

The function ψ_1 on the edge E_1 can be written in a form similar to (63)

$$\psi_1(x) = \alpha \sin k_1(x - x^*) + \beta \cos k_1(x - x^*). \tag{68}$$

Then formula (66) implies that the spectral gap decreases as the interval $[x^* - \ell/2, x^* + \ell/2]$ is cut away from the graph if

$$|\alpha| > |\beta|. \tag{69}$$

and the following estimate is satisfied

$$\cot\left(\frac{k_1\ell}{2}\right) \geq \frac{2}{k_1\mathcal{L}^*\left(\frac{\alpha^2}{\beta^2} - 1\right)}. \tag{70}$$

Using the fact that under condition (67) we have $\cot\left(\frac{k_1\ell}{2}\right) \geq \frac{\pi}{2k_1\ell}$ the following explicit estimate on ℓ can be obtained

$$\ell \leq \frac{\pi}{4}(\mathcal{L} - \ell)\left(\frac{\alpha^2}{\beta^2} - 1\right), \tag{71}$$

of course under condition (69). For the spectral gap not to increase it is enough that estimate (71) is satisfied for at least one eigenfunction ψ_1:

$$\ell \leq \min\left\{\frac{\pi}{2k_1},\ \frac{\pi}{4}(\mathcal{L} - \ell)\max_{\psi_1 : L^{\mathrm{st}}(\Gamma)\psi_1 = \lambda_1\psi_1}\left(\frac{\alpha^2}{\beta^2} - 1\right)\right\}, \tag{72}$$

where we have taken into account (67).

We see that if the eigenfunction ψ_1 is sufficiently asymmetric with respect to the point x^* (*i.e.* (69) is satisfied), then a certain sufficiently small interval can be cut from the edge ensuring that the spectral gap decreases despite the total length decreasing. Additional condition (69) was expected, since if ψ_1 is symmetric with respect to x^*, then the spectral gap may increase for any ℓ. Indeed, one may imagine that deleting of the interval is performed in two stages. One cuts the edge E_1 at the point x^* first. Then one deletes the intervals $[x^* - \ell/2, x_1^*]$ and $[x_2^*, x^* + \ell/2]$. If $\alpha = 0$ (symmetric function), then the spectral gap may be preserved in accordance to Theorem 6.7. Deleting the pending edges (intervals $[x^* - \ell/2, x_1^*]$ and $[x_2^*, x^* + \ell/2]$) may lead only to an increase of the spectral gap due to Theorem 6.5.

As was already mentioned above, spectral gap is closely related to the total length of the corresponding quantum graph. Fixing the total length and changing the topology of the underlying metric graph it becomes apparent, that the string graph has the smallest spectral gap among all graphs of the same total length [14, 24]. This is the content of the forthcoming section.

6.3 Rayleigh theorem for quantum graphs

The classical Rayleigh theorem states that the gap between the lowest two eigenvalues of the Neumann Laplacian in a domain of fixed area is maximal if

the domain is a disc. Our aim is to prove an analog of this theorem for Laplace operators on graphs with standard matching conditions at the vertices. This section is a content of the article [24] that is currently in preparation.

Theorem 6.9. *Let Γ be a connected metric graph with the total length $\mathcal{L}$ and $L^{st}(\Gamma)$ the corresponding Laplace operator defined on the domain of functions satisfying standard matching conditions at the vertices. Consider as well the graph $\Delta_{\mathcal{L}}$ formed by one interval of length $\mathcal{L}$ and the corresponding standard Laplacian $L(\Delta)$. Point $\lambda_0 = 0$ is a simple eigenvalue for both Laplacians. Then the following inequality holds for the lowest nontrivial eigenvalues:*

$$\lambda_1(\Gamma) \geq \lambda_1(\Delta).$$

Proof. Consider the eigenfunction ψ_1 corresponding to the eigenvalue $\lambda_1(\Gamma)$. This eigenvalue is a minimum of the *Rayleigh quotient*

$$\lambda_1(\Gamma) = \min_{\varphi \perp 1} \frac{\int_\Gamma |\varphi'(x)|^2 \, dx}{\int_\Gamma |\varphi(x)|^2 \, dx}, \tag{73}$$

taken over all continuous functions on Γ orthogonal to the ground state $\psi_0 \equiv 1$. The minimum is attained for $\varphi = \psi_1$ and thus the eigenfunction is a minimizer of (73).

Consider the graph Γ^*- the double cover of Γ- obtained from Γ by doubling each edge. The new edges connect the same vertices. Any function from $L_2(\Gamma)$ can be lifted to $L_2(\Gamma^*)$ in a symmetric way by assigning it the same values on the pairs of edges as on the edges of the original graph Γ. The function ψ_1^* obtained in this way obviously satisfies

$$\lambda_1(\Gamma) = \frac{\int_{\Gamma^*} |\psi_1^{*\prime}(x)|^2 \, dx}{\int_{\Gamma^*} |\psi_1^*(x)|^2 \, dx}.$$

Every vertex in Γ^* has even valency and therefore there exists a closed (Eulerian) path $\mathcal{P}$ (see [9], [15]) on Γ^* crossing each edge precisely one time. The path $\mathcal{P}$ can be identified with a loop $S_{2\mathcal{L}}$ of length $2\mathcal{L}$.

Consider now the function ψ_1^* as a function on the loop $S_{2\mathcal{L}}$. It is continuous function orthogonal to the ground state function on the loop and therefore gives an upper estimate for the corresponding eigenvalue for the Laplacian on the loop

$$\lambda_1(S_{2\mathcal{L}}) \leq \frac{\int_{S_{2\mathcal{L}}} |\psi_1^{*\prime}(x)|^2 \, dx}{\int_{S_{2\mathcal{L}}} |\psi_1^*(x)|^2 \, dx} = \lambda_1(\Gamma).$$

We obtain the result noticing that

$$\lambda_1(S_{2\mathcal{L}}) = \lambda_1(\Delta_{\mathcal{L}}),$$

that follows from (11) and (12). □

7 Conclusion

In the present thesis we focused on spectral properties of quantum graphs. As the spectra are not explicitly computable in general case we provided a numerical tool capable of solving the spectral problem for arbitrary compact graph and standard magnetic Schrödinger operator. This was done using and extending the functionality of algorithm package Chebfun which implements Chebyshev spectral methods.

Being able to compute the spectrum we started exploring related properties of quantum graphs. First, we computed the Euler characteristic (34) from the trace formula using far less eigenvalues than estimated. Next, we concentrated over the first excited eigenvalue, also called spectral gap, in particular its connection to the underlying metric graph's algebraic connectivity which has special significance as a measure of synchronizability and robustness [16]. It has been shown that dropping one edge does not necessarily mean increase of the spectral gap but that there is sufficient condition for achieving it. Also, if the new edge added to the graph is long enough (longer than the original graph itself), the spectral gap decreases according to Weyl's law.

We have shown that deleting not so long edges or cutting away short intervals from edges may lead to a decrease of the spectral gap despite the fact that total length of the graph decreases. This effect reminds us of the phenomena discovered in [10], where behavior of the spectral gap under extension of edges was discussed. It appeared that the ground state may decrease with the increase of the edge lengths, provided graphs are of complicated topology. Finally, we proved that the string has the lowest spectral gap among all graphs of the same total length.

This topic provides possibilities for further extension. As for the numerical part, Chebfun open source package implementation is the ongoing goal. Regarding the analytical part, some conjectures about the spectral gap of the standard Laplacian might extend the theory, above all the fact that while string is the graph with the lowest spectral gap, the equilateral *complete graph* should be the one with the *highest* spectral gap among the graphs with the same total length and number of vertices.

8 References

[1] Z. Battles, L. N. Trefethen, *An Extension of MATLAB to Continuous Functions and Operators*, SIAM J. Sci. Comput., Vol. 25, No. 5, 2004, pp. **1743-1770**.

[2] G. Berkolaiko, P. Kuchment,*Dependence of the spectrum of a quantum graph on vertex conditions and edge lengths*, Spectral geometry, 117137, *Proc. Sympos. Pure Math.*, **84**, Amer. Math. Soc., Providence, RI, 2012.

[3] G. Berkolaiko, P. Kuchment, *Introduction to quantum graphs*, *Mathematical Surveys and Monographs*, 186. American Mathematical Society, Providence, RI, 2013.

[4] J. P. Boyd, *A Numerical Comparison of Seven Grids for Polynomial Interpolation on the Interval*, *Computers and Mathematics w. Applications*, **38** (1999) 35-50.

[5] C. Canuto, M. Y. Hussaini, A. Quarteroni, T. A. Zang, *Spectral Methods: Fundamentals in Single Domains*, Springer-Verlag, Berlin, Heidelberg, 2006.

[6] C. Cattaneo, *The spectrum of the continuous Laplacian on a graph*, *Monatsh. Math.* **124** (1997), no. 3, 215-235.

[7] F. Chung, Spectral graph theory, CBMS Regional Conference Series in Mathematics, 92, American Mathematical Society, Providence, RI, 1997.

[8] F. Bornemann, T. A. Driscoll, L. N. Trefethen, *The Chebop System for Automatic Solution of Differential Equations*, BIT Numerical Mathematics (2008) **48**: 701723.

[9] L. Euler, *Solutio problematis ad geometriam situs pertinentis*, *Comment. Academiae Sci. I. Petropolitanae* 8 (1736), **128-140**.

[10] P. Exner, M. Jex, *On the ground state of quantum graphs with attractive δ-coupling*, *Phys. Lett.* A **376** (2012), no. 5, 713-717.

[11] P. Exner, O. Post, *Quantum Networks Modeled by Graphs*, Proceedings of the Joint Mathematics/Physics Workshop "Quantum Few-Body System" (Aarhus 2007), AIP Conf. Proc., vol. 998; Melville, NY, 2008, pp. **1-17**.

[12] P. Exner and P. Šeba, *Free quantum motion on a branching graph*, Rep. Math. Phys., 28 (1989), **7-26**.

[13] M. Fiedler, *Algebraic Connectivity of Graphs*, Czechoslovak Mathematical Journal, 23 (98), 1973, Praha, **298-305**.

[14] L. Friedlander, *Extremal properties of eigenvalues for a metric graph*, *Ann. Inst. Fourier* **55** (2005), no. 1, 199211.

[15] C. Hierholzer, Chr. Wiener, *Über die Möglichkeit, einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren*, *Math. Ann.* **6** (1873), no. 1, **30-32**.

[16] M. Holroyd, *Connectivity and Synchronizability of Discrete Complex Systems*, 2006, Williamsburg, VA, Honors thesis.

[17] W. Huang, D. M. Sloan, *The Pseudospectral Method for Third-Order Differential Equations*, SIAM Journal on Numerical Analysis, Vol. 29, 1992, **1626-1647**.

[18] V. Kostrykin, R. Schrader *Laplacians on Metric Graphs: Eigenvalues, Resolvents and Semigroups*, Quantum Graphs and Their Applications, Contemp. Math., Am. Math. Soc., Providence, 2006, pp. **201-225**.

[19] P. Kuchment, *Quantum Graphs I: Some Basic Structures*, Waves Random Media, 14, **107-128**, 2004.

[20] P. Kurasov, *Inverse Problems For Quantum Graphs: Recent Development and Perspectives*, Acta Physica Polonica A, 120 , v6A (2011), **132-141**.

[21] P. Kurasov, *Quantum Graphs: Spectral Theory and Inverse Problems*, to appear.

[22] P. Kurasov, *Schrödinger operators on graphs and geometry. I. Essentially bounded potentials*, *J. Funct. Anal.*, **254** (2008), no. 4, 934–953.

[23] P. Kurasov, G. Malenova, S. Naboko, *On the spectral gap for quantum graphs*, to appear in J. Phys. A: Math. Theor.

[24] P. Kurasov and S. Naboko, On Rayleigh theorem for quantum graphs, in preparation.

[25] G. Meurant and Z. Strakos, *The Lanczos and conjugate gradient algorithms in finite precision arithmetic*, Acta Numerica, 15, Cambridge University Press, 2006, pp. **471-542**.

[26] K. Pankrashkin, *Spectra of Schrödinger Operators on Equilateral Quantum Graphs*, Letters in Math. Physics (2006) 77: **139-154**.

[27] O. Post, *Spectral Analysis on Graph-like Spaces*, Springer Lecture Notes Vol **2039**, 431p, 2012.

[28] L. N. Trefethen, *Spectral Methods in MATLAB*, SIAM, Philadelphia (2000).

[29] J. A. C. Weideman, S. C. Reddy, *A MATLAB Differentiation Matrix Suite*, ACM Transactions on Mathematical Software, Vol. 26, No. 4, 2000, **465-519**.

9 Appendix

@GRAPH class

```
function s = char(A,name)
% CHAR  Convert graph to pretty-printed string.

% extract information about the graph
    matrix = A.matrix;
    m = A.nume;
    n = A.numv;
    len = A.lengths;
    s = '';

    if nargin == 1
        name = '';
    end

    short = isequal(get(0,'Format'),'short');
    if short == 3
        formatSpec = '%10.4f';
    else
        formatSpec = '%10.15f';
    end
    % matrix format
        s = char(s, sprintf('   %sx%s graph defined by incidence matrix:'...
            ,int2str(m),int2str(n)));
        s = char(s, int2str(matrix));
        s = char(s, sprintf('   with %d edges and %d vertices',m,n));
        s = char(s, sprintf('   and edge lengths: %s',num2str(len)));
end
```

```
function D = diff(d,m,varargin)
% DIFF Differentiation operator.
% D = DIFF(R) returns a chebop representing differentiation for chebfuns
% defined on the graph R.
%
% D = DIFF(R,M) returns the chebop for M-fold differentiation.
% D = DIFF(R,0) returns EYE(R)
%
ne = d.nume;    % number of edges

if nargin == 1, m = 1; end      % 1 is default order if none specified

```

```
13  if round(m)~=m
14      error('CHEBFUN:domain:diff:fracdiff',...
15          'Fractional derivatives are not yet supported as operators.');
16  end
17
18  % 0th derivative -> identity operator
19  if m == 0
20      D = eye(d);
21  % no domain
22  elseif isempty(d)
23      D = linop;
24      return
25  % matrix is non-empty
26  else
27      % setup domains
28      len = d.lengths;        % extract edge lengths
29      dd = [-len/2; len/2];   % symmetric wrt 0
30      dd = dd';
31  % pre-allocate domain cell
32      dom = cell(ne,1);
33          for kk = 1:ne
34              dom{kk} = domain(dd(kk,:));
35          end
36
37      for ii = 1:ne           % zero chebop on off-diagonals
38          for jj = 1:ne
39              D.blocks{ii,jj} = zeros(dom{jj});
40          end
41      end
42      for ii = 1:ne           % differentiation matrices on diagonal
43          D.blocks{ii,ii} = diff(dom{ii},m);
44      end
45      D.graph = d.matrix;     % remember the incidence matrix
46      D.domain = [];          % domain is empty
47  end
48  end
```

```
1  function display(A)
2  % DISPLAY Pretty-print a graph.
3  %
4  % DISPLAY is called automatically when a statement that results in a
5  % chebarray output is not terminated with a semicolon.
6
7  % Copyright 2011 by The University of Oxford and The Chebfun Developers.
```

```
% See http://www.maths.ox.ac.uk/chebfun/ for Chebfun information.

    loose = ~isequal(get(0,'FormatSpacing'),'compact');
    if loose, disp(' '), end

    disp([inputname(1) ' = ']);

    s = char(A,inputname(1));
    if ~loose
        s( all(isspace(s),2), : ) = [];  % Remove blank lines
    end
    disp(s)

    if loose, disp(' '), end

end
```

```
function I = eye(d)
% EYE Identity operator.
% EYE(G) returns a cell of chebops representing the identity for functions
% defined on the graph G.
%
% setup domains
    len = d.lengths;         % extract edge lengths
    dd = [-len/2; len/2];    % domain symmetric wrt 0
    dd = dd';
    ne = d.nume;             % number of edges

% pre-allocate domain cell
    dom = cell(ne,1);

if isempty(d)
    I = linop;
else
    % cell of domains
        for kk = 1:ne
            dom{kk} = domain(dd(kk,:));
        end
    % off-diagonal entries are empty linops
    for ii = 1:ne
        for jj = 1:ne
            I.blocks{ii,jj} = zeros(dom{jj});
        end
```

```
    end
    % digonal cells are identity matrices
    for ii = 1:ne
        I.blocks{ii,ii} = eye(dom{ii});
    end
    I.graph = d.matrix;     % remember the incidence matrix
    I.domain = [];          % domain is empty
end
end
```

```
function D = getpotential(d,varargin)
% GETPOTENTIAL transforms the electric potential Q given by user
% (anonymous function or cell) to linop. Laplacian is in the form
% L = -d^2/dx^2 + Q
% Each edge is a subset [-l/2,l/2].
%
% D = getpotential(D,Q) returns a linop representing potential Q for
% chebfuns defined on the graph D.
%
% Example:
% t = graph([-1 1 0 0; 0 -1 1 0; 0 -1 0 1]');   % star-like domain
% q{1} = @(x,u) 20*x.^5.*u;             % potential on the first edge
% q{2} = @(x,u) cos(30*x).*u;           % on the second edge
% q{3} = @(x,u) 20*exp(-x.^3).*u;       % on the third edge
% gpot = getpotential(t,q);             % cell of linops
%
%
ne = d.nume;            % number of edges

if nargin == 1          % no potential given
    q = @(x,u) 0.*x.*u;
    D = getpotential(d,q);
    return;
end;

if nargin == 2          % same potential for all edges
    pote = varargin{:}; % concatenate input
    if numel(pote) == 1
        pote = repmat(varargin,1,ne);   % repeat if the user has been lazy
    end
end

if numel(pote)~= ne
    error('CHEBFUN:graph:getpotential',...
```

```
            'Wrong number of potentials given.');
end

% no domain
if isempty(d)
    D = linop;
    return
% matrix is non-empty
else
    % setup domains
    len = d.lengths;         % extract edge lengths
    dd = [-len/2; len/2];    % choose domains symmetric wrt 0
    dd = dd';                % transpose

% pre-allocate domain cell
    dom = cell(ne,1);
        for kk = 1:ne
            dom{kk} = domain(dd(kk,:));
        end
    for ii = 1:ne
        for jj = 1:ne
           D{ii,jj} = zeros(dom{jj});   %#ok<AGROW>
        end
    end
    for ii = 1:ne
        f = chebop(dd(ii,:));       % empty chebop on correct domain
        f.op = pote{ii};
        D{ii,ii} = linop(f);    %#ok<AGROW> % linearize chebop to linop
    end
end
end
```

```
classdef (InferiorClasses = {?double}) graph

    properties
        matrix = [];         % incidence matrix
        nume = [];           % number of edges
        numv = [];           % number of vertices
        lengths = [];        % vector of edge lengths
    end

    methods
        function t = graph(varargin)

```

```
13              % Parse the inputs
14              if nargin == 0
15                  % Return an empty graph
16                  return
17              elseif nargin == 1 && isa(varargin{1},'graph')
18                  % Return input graph
19                  t = varargin{1};
20                  return
21              end
22
23              % first argument defines the matrix
24                  m = varargin{1};
25                  [nv,ne] = size(m);
26                  t.nume = ne;
27                  t.numv = nv;
28
29              % second variable defines the edge lengths
30                  if nargin >= 2
31                      len = varargin{2};
32                  else len = 2*ones(1,ne);
33                  end
34              % repeat if the user has been lazy
35                  if numel(len) == 1
36                      len = repmat(len,1,ne);
37                  end
38                  t.lengths = len;
39
40              % validate the length vector
41                  if numel(t.lengths)~=t.nume
42                      error('CHEBFUN:graph',...
43          'Length vector does not have correct number of elements')
44                  end
45
46              % setup the matrix
47                  t.matrix = m;
48
49              % validate the input
50                  ish = validate(t);
51              if ~ish
52                  error('CHEBFUN:graph', 'Incidence matrix mismatch.');
53              end
54          end
55      end
56  end
```

```
function [V,D] = grapheig(G, numEigs, varargin)
% GRAPHEIG(G, numEigs) computes spectrum of Laplace operator on
% a graph given by its incidence matrix and edge lengths. There is
% a continuity condition imposed at the junctions and Neumann boundary
% condition is default on the pending vertices. numEigs defines the
% number of eigenvectors and eigenvalues required.
%
% GRAPHEIG(G, numEigs, bcD) assingnes the vertices given by bcD with
% Dirichlet boundary condition. The remaining pending vertices are
% endowed with Neumann boundary condition.
%
% GRAPHEIG(G, numEigs, bcD, bcN) assignes pending vertices bcD
% with Dirichlet boundary conditions and bcN with Neumann boundary
% conditions respectively.
%
% GRAPHEIG(G, numEigs, bcD, bcN, q) adds electric potential to the
% Laplace operator L = diff^2 + q. Define the potential as anonymous
% function on each edge, mind the edge parametrization [-len/2,len/2]
%
% GRAPHEIG(G, numEigs, bcD, bcN, q, a) add magnetic potential, the
% operator is now in the form L = (i*diff + a)^2 + q. Define the magnetic
% potential as an anonymous function on each edge. The potential is
% required to be smooth.
%
% EXAMPLE
% A = [-1 1 0 0; 0 1 -1 0; -1 0 1 0; 0 0 1 -1]';
% len = [1,2,3,4];
% G = graph(A,len);
% q = @(x,u) 0.*x.*u;
% a = @(x) 1 + 2.*x.^2;
% [V,D] = grapheig(G,4,[],[4], q, a)

% extract information about graph
A = G.matrix;
len = G.lengths;
nv = G.numv;
ne = G.nume;

% find which indicies are pending vertices or junctions
pen = find(sum(abs(A),2) == 1);
jun = 1:nv;
jun(pen) = [];
npen = numel(pen);  % number of pending vertices

% Neumann BC and zero potentials are default setting
    q = @(x,u) 0.*x.*u;
```

```
47      bcD = [];
48      bcN = pen;
49      mag = @(x) 0.*x;
50  % check for varargin
51  if numel(varargin) >= 1 && isa(varargin{1},'double')
52      bcD = varargin{1};
53  % if only one BC set defined, the other is complement
54      bcN = setxor(pen,bcD);
55  end
56  % both Dirichlet and Neumann are defined
57  if numel(varargin) >= 2 && isa(varargin{2},'double')
58      bcN = varargin{2};
59  end
60  % electric potential
61  if  numel(varargin) >= 3 && (isa(varargin{3},'function_handle')...
62          || isa(varargin{3},'cell'))
63      q = varargin{3};
64  end
65  % magnetic potential
66  if numel(varargin) == 4 && (isa(varargin{4},'function_handle')...
67          || isa(varargin{4},'cell'))
68      mag = varargin{4};
69  end
70
71  % boundary conditions for pending edges (note, always homogeneous)
72  bcN_val = zeros(1,numel(bcN)); % Neumann values
73  bcD_val = zeros(1,numel(bcD)); % Dirichlet values
74
75  % check for number of BC inputs
76  if numel([bcD bcN]) > npen
77      error('CHEBFUN:graph:diff',...
78          'Too many boundary condition inputs.');
79  end
80
81  if numel([bcD bcN]) < npen
82      error('CHEBFUN:graph:diff',...
83          'Not enough boundary condition inputs.');
84  end
85
86  % BC's not allowed at junctions
87  if any(intersect([bcD bcN],jun))
88      error('CHEBFUN:graph',...
89          'Boundary condition must not be defined at the junction.');
90  end
91
92  %% build differential operator
```

```
93      graphD2 = diff(G,2);                    % structure
94  % build electric potential operator
95      graphP = getpotential(G,q);             % cell of linops
96  % add potential to the operator
97  for i = 1:numel(graphD2.blocks)
98      graphD2.blocks{i} = graphD2.blocks{i} + graphP{i};
99  end
100
101 D2 = chebarray(graphD2);                  % build linop
102 Akeep = A;                                % store the incidence matrix
103
104 %% initialise bcvals (for rhs)
105 bcvals = zeros(2*ne,1);
106
107 % initialise domains
108 d = zeros(2,ne);
109 for o = 1:ne
110     d(:,o) = D2.blocks{o,o}.domain;   % domain
111 end
112  %% MAGNETIC POTENTIAL
113  if numel(mag) == 1
114     a = cell(ne,1);
115     for mm = 1:ne
116         a{mm} = mag;   % repeat if the user has been lazy
117     end
118 end
119 % get magnetic fluxes
120 flux = zeros(ne,1);     % initiate
121 for ii = 1:ne
122     flux(ii) = sum(chebfun(a{ii},d(:,ii)));% integrate over the edge
123 end
124     flux = exp(1i*flux);
125
126 % check for correct number of magnetic potentials
127 if numel(a) ~= ne
128     error('CHEBFUN:graph','Wrong number of magnetic potentials')
129 end
130 %% Initiate boundary and matching conditions
131 % Dirichlet + pending edge
132 npen_d = numel(bcD);
133 ajD = A(bcD,:);                       % respective incidence matrix columns
134 bcvals(1:npen_d) = bcD_val;           % restore bc values
135
136 % Neumann + pending edge
137 npen_n = numel(bcN);
138 ajN = A(bcN,:);
```

```
139 bcvals(npen_d+(1:npen_n)) = bcN_val;
140
141 % Neumann + junction
142 ajJ = A(jun,:);
143
144 % Dirichlet + junction
145 ndir = sum(sum(abs(A),2) - 1);      % number of dirichlet conditions
146
147 % initialise
148 bc_pen_d = [];
149 bc_pen_n = [];
150 bc_jun_d = [];
151 bc_jun_n = [];
152
153 % initiate loops
154 l = 1;
155 m = 1;
156 n = 1;
157 o = 1;
158
159 % chebarray block array does not support different domains
160 for ii = 1:ne
161     d2 = d(:,ii);
162     d2 = domain(d2);
163     fr = feval(d2,1);       % eval at 1 (right)
164     zr = fr*zeros(d2);      % zero row
165     bc_jun_d{ii} = zr;      %#ok<AGROW>
166 end
167 bc_jun_d = repmat(bc_jun_d,ndir,1);
168
169 for jj = 1:nv                   % loop through all vertices
170     %% Dirichlet condition for pending vertices
171     if any(jj == bcD)               % hit the pending edge with Dir BC
172         % setup operators
173         j = find(A(jj,:));          % index of the edge
174         d1 = domain(d(:,j));        % respective domain
175         for k = 1:ne
176             alk = ajD(l,k);
177             eend = d1.ends;             % edge ends
178             eend = alk*abs(eend(1));    % retrieve the sign
179                 % absolute value to get rid of 0
180             bc_pen_d{l,k} = abs(alk)*feval(d1,eend); %#ok<AGROW>
181         end
182         l = l + 1;
183     end
184
```

```
185      %% Neumann condition for pending vertices
186      if any(jj == bcN)                % hit the pending edge with Neu BC
187          % setup operators
188          j = find(A(jj,:));           % index of the edge
189          d1 = domain(d(:,j));         % respective domain
190          df = diff(d1);               % differential operator on domain
191          for k = 1:ne
192              alk = ajN(m,k);
193              eend = d1.ends;
194              eend = alk*abs(eend(1));
195              bc_pen_n{m,k} = alk*feval(d1,eend)*df;  %#ok<AGROW>
196          end
197          m = m + 1;
198      end
199
200      %% neuman condition at junction
201      if any(jj == jun)                % hit the junction with Neu BC
202          for k = 1:ne
203              d1 = domain(d(:,k));     % respective domain
204              df = diff(d1);           % differential operator on domain
205              alk = ajJ(n,k); % determines sign and where we evaluate
206              eend = d1.ends;
207              eend = alk*abs(eend(1));
208               if alk == 1             % magnetic potential if right endpoint
209                   alk = alk*flux(k);
210               end
211              bc_jun_n{n,k} = alk*feval(d1,eend)*df;  %#ok<AGROW>
212          end
213          n = n + 1;
214      end
215
216      %% dirichlet condition at junction
217      if any(jj == jun)
218          k = find(A(jj,:),1);
219          ajk = A(jj,k);
220          A(jj,k) = 0;                      % erase nonzeros
221          while any(A(jj,:))
222              d1 = domain(d(:,k));         % respective domain
223              eend = d1.ends;              % retrieve endpoints
224              eend = ajk*abs(eend(1));
225               if ajk == 1                 % add magnetic potential
226                  bc_jun_d{o,k} = flux(k) * feval(d1,eend);
227               else
228                   bc_jun_d{o,k} = feval(d1,eend);
229               end
230
```

```
231             k = find(A(jj,:),1);
232             d1 = domain(d(:,k));        % respective domain
233             ajk = A(jj,k);
234             A(jj,k) = 0;
235             eend = d1.ends;
236             eend = ajk*abs(eend(1));
237                     if ajk == 1         % magnetic potential if right
238                         bc_jun_d{o,k} = -flux(k) * feval(d1,eend);
239                     else
240                         bc_jun_d{o,k} = -feval(d1,eend);
241                     end
242             o = o+1;
243         end
244     end
245 end
246
247 %% combine
248 bc = [bc_pen_d ; bc_pen_n ; bc_jun_n ; bc_jun_d];
249 % create struct to keep the graph information
250 chebbc = {};
251 chebbc.blocks = bc;
252 chebbc.domain = [];
253 chebbc.graph = Akeep;
254
255 % retrieve chebarray
256 chebbc = chebarray(chebbc);
257 D2.bc = chebbc;
258 D2.bcvals = bcvals;
259 D2.lengths = len;
260
261 % compute eigenvalues
262 [V D] = eigs(-D2,numEigs,'SM');
263 D = sort(diag(D));
```

```
1 function e = isempty(A)
2 %ISEMPTY   True for empty graph.
3
4 e = isempty(A.matrix);
5
6 end
```

```
1 function e = isnan(A)
2 %ISNAN   True for NaN.
```

```

e = isnan(A.matrix);
end
```

```
function out = subsref(A,t)
% SUBSREF  Retrieve information from a graph.
%
% SUBSREF can be used to call:
%   '.MATRIX'    - The incidence matrix
%   '.NUME'      - Number of edges
%   '.NUMV'      - Number of vertices
%   '.LENGTHS'   - Vector consisting of edge lengths
%

switch t(1).type

    case '.'
        switch t(1).subs
            case 'matrix'
                out = A.matrix;
            case 'nume'
                out = A.nume;
            case 'numv'
                out = A.numv;
            case 'lengths'
                out = A.lengths;
            otherwise
                error('CHEBFUN:chebarray:subsref:unknown', ...
                    'Unknown property: %s.',t(1).subs);
        end
        if numel(t) >1
            out = subsref(out,t(2:end));
        end
end
end
```

```
function ish = validate(A)
% VALIDATE   Validate the incidence matrix
%
% ISH = VALIDATE(A) validates the domain and dimensions of a graph.

    mat = A.matrix;
    ish = true;
```

```
    % check whether all elements are +-1,0
    for ii = 1:A.nume
         edge = nonzeros(mat(:,ii));
            if ~(edge == [-1; 1] | edge == [1; -1])
                ish = false;
            end
    end
end
```

MAIN DIRECTORY

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% DRAWGRAPH This allows the user to draw a graph on the plain canvas,
% boundary conditions may be set by hand. After the metric graph is
% drawn, its incidence matrix and edge lengths are computed and given
% to grapheig.m and plottree.m to compute and plot the resulting
% eigenvectors and eigenvalues.
% Neumann conditions are default in case the user forgets to define
% it. Isolated points are treated as useless and are removed from the
% computations.
% Follow the instructions in the figure title.

% number of eigenvectors required
numEigs = 4;

axis([0 10 0 10])
hold on
% Initially, the list of points is empty.
xy = [];
n = 0;

% Loop, picking up the points.
but = 1;
figure(1);
title('Pick the nodes. Right mouse button picks the last point.',...
    'FontSize',16)
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'o')
    n = n+1;
    xy(:,n) = [xi;yi];      %#ok<AGROW>
end

% the list of edges is empty
```

```
ed = [];
but2 = 1;
title('Connect points by edges. Right mouse click marks the last point.',...
    'FontSize',16)
% Plot the lines inbetween the points
while but2 == 1
    % pick starting point
   [xi,yi] = ginput(1);
   [lend,inl] = findclosest(xy,[xi;yi]);    % find closest point
   plot(lend(1),lend(2),'mo')               % indicate the selection

   % pick ending point
   [xi2,yi2,but2] = ginput(1);
   [rend,inr] = findclosest(xy,[xi2;yi2]);  % find closest point
   plot(rend(1),rend(2),'mo')               % indicate the selection

   ts = [linspace(lend(1),rend(1),36);...
       linspace(lend(2),rend(2),36)];       % walk the line

% plot the interpolating line
   plot(ts(1,:),ts(2,:),'k')
% and make the points blue again
   plot(rend(1),rend(2),'o')
   plot(lend(1),lend(2),'o')

% check whether the edge is not degenerated
if ~all(lend == rend)
% save the endpoints along with their indices
ed = [ed, [lend; rend; inl; inr]];          %#ok<AGROW>
end
end

% Drop the double edges
ed = unique(ed', 'rows');
ed = ed';

% number of edges and vertices
ne = size(ed,2);
nv = size(xy,2);

% retrieve incidence matrix and edge lengths
len = sqrt((ed(1,:) - ed(3,:)).^2 + (ed(2,:) - ed(4,:)).^2);
% initialize incidence matrix
incm = zeros(ne, nv);
for i = 1:ne
   incm(i,ed(5:6,i)) = [-1,1];
```

```
end

% define the boundary conditions- either Neumann or Dirichlet
% if not specified, Neumann is default
ede = ed(5:6,:);                % edge endpoints
numoc = histc(ede(:)',1:nv);    % number of occurence of each endpoint
neu = find(numoc == 1);         % those occuring once are pending vertices
dir = [];

% initiate counter
but3 = 1;

title('Set BC. Neumann by left-click (green), Dirichlet by right-click (red).',.
    'FontSize',16)
while but3 == 1 || but3 == 3
% pick starting point
   [xi,yi,but3] = ginput(1);
   [l,in] = findclosest(xy,[xi;yi]);    % find closest point
   title('Boundary conditions. Press any key when done.','FontSize',16)
% Neumann case when left-clicking
   if but3 == 1
        plot(l(1),l(2),'go')            % indicate the selection
% check if it is not a junction
     if numoc(in) > 1                   % number of occurance
      error('Boundary condition must not be defined at the junction.')
     end
   neu = [neu in];                      %#ok<AGROW>
% remove from dirichlet set in case it already has been labeled
   dir = setdiff(dir,in);

% Dirichlet when right-click
   elseif but3 == 3
          plot(l(1),l(2),'ro')          % indicate the selection
% check if it is not a junction
     if numoc(in) > 1                % number of occurance
      error('Boundary condition must not be defined at the junction.')
     end
   dir = [dir in];                      %#ok<AGROW>
% remove from neu set in case it already has been labeled
   neu = setdiff(neu,in);
   end
end

% in case the user labeled one vertex more times
neu = unique(neu);
dir = unique(dir);
```

```
126
127 % remove all isolated points
128 rem = find(numoc == 0);      % isolated points
129 neu = setdiff(neu,rem);      % remove it is has been assigned by Neumann bc
130 dir = setdiff(dir,rem);      % same for Dirichlet
131 % shift the node numbering
132 iso = numoc == 0;
133 iso = cumsum(iso);           % shifts for each node
134 if ~isempty(neu)             % if there is any neumann vertex
135     neu = neu - iso(neu);
136 end
137 if ~isempty(dir)             % if there is any dirichlet vertex
138     dir = dir - iso(dir);
139 end
140
141 incm(:,rem) = [];            % remove from incidence matrix
142 xy(:,rem) = [];              % remove from the set of edges
143
144 % assemble the graph
145 G = graph(incm',len);
146
147 % generate graph and compute eigenvalues/vectors
148 [V,D] = grapheig(G, numEigs, dir, neu);
149
150 % plot the result
151 plottree(V,incm',xy(1,:) + 1i*xy(2,:))
152 title('First few eigenvectors','FontSize',16)
153 shg
```

```
1 function [xx,in] = findclosest(xset,x)
2 % FINDCLOSEST
3 % Given a set of points xset and a simple point x, we compute the distance
4 % to the original set, pick up the closest point and return its
5 % coordinates and index
6
7    [dist,in] = min((xset(1,:) - x(1)).^2 + (xset(2,:) - x(2)).^2);
8    xx = xset(:,in);
9 end
```

```
1 function plottree(f,A,v,varargin)
2 % PLOTTREE(F,A,V) plots the chebarray/quasimatrix F on the directed
3 % tree/graph defined by the adjacency matrix A and vertices V
4 % (complex-valued). If V is not given, the vertices are equaispaced around
```

```
% the unit circle.

% If we're not given an adjacency matrix, just plot the chebarray
if nargin < 2
    plot(f)
    return
end

% Count the number of edges and vertices
nv = size(A,1);
ne = size(A,2);

% Hold the plot?
ish = ishold;
% Clear the axis if it's not.
if ~ish
    cla(gca, 'reset')
end

% make up some vertices if none are given
if nargin < 3
    v = exp(2*pi*1i*linspace(0,1,nv+1));
end

if min(size(f)) > 1
    hold on
    for k = 1:size(f,2)
        g = f;
        g = g(:,k);
        plottree(g,A,v,varargin{:});
    end
% deal with 'hold on'
    if ~ish, hold off, end
    return
end

% setup
d =  domain(f);
cols = get(0,'DefaultAxesColorOrder');
cols = repmat(cols,5,1);

for k = 1:ne
    int = d([k k+1]);        % k-th particular interval
    a = v( A(:,k)==-1 );     % start of edge
    b = v( A(:,k)==1  );     % end of edge
    x = chebfun([a b],int);   % a linear chebfun
```

```
51      ek = f(int);            % the kth edge
52  % plot the vertices
53      if isnumeric(ek)
54          vals = ek;
55      else
56          vals = [get(ek.funs(1),'lval') get(ek.funs(end), 'rval')];
57      end
58      plot3(real([a b]),imag([a b]),vals,'ok',varargin{:}); hold on
59      % plot the edge
60      plot3(real(x),imag(x),ek,'color',cols(k,:),varargin{:});
61  end
62
63  view(-10,20);
64
65  % deal with 'hold on'
66  if ~ish
67      hold off
68  end
```