NUMERICAL METHODS FOR SPECTRAL THEORY

FELIX KWOK

1. Introduction

The purpose of these notes is to introduce a few numerical methods for approximating the eigenvalues and eigenfunctions of partial differential operators. Such numerical approximations are often important in physics and engineering, as they can be used to describe the vibrations of a physical structure.

1.1. Vibrations and Eigenvalues.

Consider a thin membrane modelled as a two-dimensional bounded, open subset Ω with boundary $\partial\Omega$. We describe its vibrating motion over a time interval $(0, t_f)$ in terms of its vertical displacement $z : \Omega \times (0, t_f) \to \mathbb{R}$, starting from an initial displacement $z_0(x, y)$ and initial velocity $v_0(x, y)$. Assume that the boundary of the membrane is fixed, so that its displacement is zero at all times. If there are no external forces acting on the membrane, i.e., the only forces driving the vibrating motion are the restoring force due to the deformation of the membrane, then Newton's second law states that

$$\frac{\partial^2 z}{\partial t^2} = -\mathscr{L} z, \qquad \text{in } \Omega \times (0, t_f],$$

where $\mathscr{L}$ is a spatial differential operator acting on $z(x, y, t)$. The precise form of $\mathscr{L}$ depends on the model used for the material and can often be derived from energy considerations; see Section 5 for the vibrating plate. For the vibrating membrane, a simple model uses the scaled *Laplacian operator*

$$\mathscr{L} z = -c\Delta z := -c\Big(\frac{\partial^2 z}{\partial x^2} + \frac{\partial^2 z}{\partial y^2}\Big), \qquad c > 0,$$

which is a second-order linear elliptic differential operator. As a result, we get the linear initial-boundary value problem

$$\begin{aligned} &\frac{\partial^2 z}{\partial t^2} = c\Delta z \qquad \text{in } \Omega \times (0, t_f) \\ &z|_{t=0} = z_0(x, y), \qquad \left.\frac{\partial z}{\partial t}\right|_{t=0} = v_0(x, y) \\ &z|_{\partial\Omega} = 0, \qquad t \in (0, t_f). \end{aligned} \tag{1}$$

The requirement that z vanish along the boundary is known as a *homogeneous Dirichlet boundary condition* on z.

1.2. Separation of Variables.

A *standing wave* is any solution of (1) of the form

$$z(x, y, t) = u(x, y) \cdot T(t).$$

Because of linearity, any linear combination of standing waves is also a solution of (1). A classical technique for solving linear problems of this type is to first seek a family of standing wave solutions, and then find a linear combination of these

solutions that satisfies the initial conditions. This procedure is known as *separation of variables*.

More concretely, if $z(x,y,t) = u(x,y)T(t)$ is a standing wave solution, then substituting into the PDE yields

$$T''(t)u(x,y) = -\mathscr{L}u(x,y)T(t)$$

or

$$-\frac{T''(t)}{T(t)} = \frac{\mathscr{L}u(x,y)}{u(x,y)}, \tag{2}$$

where we assumed that $u(x,y)$ and $T(t)$ do not vanish identically. Since the left-hand side is independent of x and y, and the right-hand side independent of t, we see that both sides are in fact constant; in particular, we have

$$\frac{\mathscr{L}u(x,y)}{u(x,y)} = \lambda \qquad \text{or} \qquad \mathscr{L}u = \lambda u, \tag{3}$$

i.e., u is an eigenfunction of $\mathscr{L}$ with eigenvalue λ, which are assumed to exist.[1] Moreover, we see that λ is non-negative, since we can multiply (3) by u and integrate to obtain

$$\int_\Omega \lambda u^2\,dx = -c\int_\Omega u\Delta u\,dx = c\int_\Omega |\nabla u|^2\,dx,$$

where the boundary terms in Green's identity vanishes because $u|_{\partial\Omega} = 0$. Thus, we see that

$$\lambda = \frac{c\int_\Omega |\nabla u|^2\,dx}{\int_\Omega u^2 dx} \geq 0.$$

Let $u_n(x,y)$ be an eigenfunction with eigenvalue λ_n. Then (2) implies that the corresponding time-dependent function $T = T_n(t)$ satisfies

$$T_n''(t) = -\lambda T_n(t) \implies T_n(t) = a_n\cos(\sqrt{\lambda_n}t) + b_n\sin(\sqrt{\lambda_n}t).$$

Thus, the eigenvalues λ_n can be interpreted as the square of the frequencies of vibration. Taking linear combinations over all such frequencies, we can write the solution $z(x,y,t)$ as

$$z(x,y,t) = \sum_{n=1}^\infty a_n\cos(\sqrt{\lambda_n}t)u_n(x,y) + b_n\sin(\sqrt{\lambda_n}t)u_n(x,y),$$

where the coefficients a_n and b_n are determined by the initial conditions

$$z(x,y,0) = \sum_{n=1}^\infty a_n u_n(x,y) = z_0(x,y),$$

$$\frac{\partial z}{\partial t}(x,y,0) = \sum_{n=1}^\infty \sqrt{\lambda_n}b_n u_n(x,y) = v_0(x,y).$$

Since the eigenfunctions $\{u_n\}_{n=1}^\infty$ form an orthonormal basis of $L^2(\Omega)$ by the spectral theorem [31, Theorem 4.1], it is always possible to find a_n and b_n such that the above equality holds in the sense of L^2-convergence for every $z_0, v_0 \in L^2(\Omega)$.

For special geometries and boundary conditions, it is possible to solve the eigenvalue problem analytically. For example, if $\Omega = (0,1)\times(0,1)$ is the unit square

[1] For the existence and uniqueness of eigenvalues and eigenvectors for the Laplacian operator with Dirichlet boundary conditions, see Chapter 5 of Richard Laugesen's notes on the spectral theory of PDEs [31].

and $u = 0$ on the boundary $\partial\Omega$, then one can again use separation of variables and find that the eigenfunctions of the negative Laplacian $\mathscr{L} = -\Delta$ (with $c = 1$) are of the form

$$u_{mn}(x, y) = \sin(m\pi x)\sin(n\pi y),$$

with corresponding eigenvalues $\lambda_{mn} = \pi^2(m^2 + n^2)$. Analytical solutions are also available for rectangle, discs and sectors [40, Chapter 11]. For more complicated geometries, however, one can only obtain approximations to the eigenvalues by replacing the PDE eigenvalue problem by a finite-dimensional, approximate matrix eigenvalue problem. As with any approximation, we are interested in the following questions:

(1) How to approximate the continuous problem by a discrete one?
(2) How good is the approximation?
(3) How to solve the associated finite dimensional problem?

To answer the first question, we will consider two methods in these notes: the finite difference method will be discussed in Section 2, and the finite element method will be discussed in Section 3. For each method, we will study its approximation properties and establish some convergence results for the Laplace problem. We will briefly consider the third question in Section 4, where we explain the fundamentals of numerical methods for matrix eigenvalue problems. Finally, in Section 5, we consider a more complicated example with a fourth order biharmonic operator and different types of boundary conditions. Section 6 provides some suggestions for further reading, and Section 7 contains practice problems given during the CRM Summer School on Spectral Theory and Applications, held in Quebec City, Canada in August 2016.

2. Finite Difference Methods

The finite difference method is perhaps the simplest discretization that can be used to approximate a PDE eigenvalue problem. In essence, one uses a Taylor-based approximation to replace partial derivatives by partial differences, and then solves the transformed eigenvalue problem, which is now finite dimensional. As we will see, the method works best for domains whose boundaries align with a rectangular grid.

2.1. Finite Difference for the 2D Laplacian Operator.

Consider the eigenvalue problem on the unit square $\Omega = (0, 1) \times (0, 1)$

$$-\Delta u = \lambda u, \qquad u = 0 \text{ on } \partial\Omega$$

Instead of seeking $u(x, y)$ everywhere, we seek an approximation of it on a *uniform grid*

$$u_{ij} \approx u(x_i, y_j), \qquad 1 \leq i, j, \leq n - 1$$

with $x_i = ih$, $y_j = jh$, $h = 1/n$. This $\{u_{ij}\}_{i,j=0}^{N}$ is called a *grid function*, since it is a function defined on the grid $\{(x_i, y_j)\}_{i,j=0}^{N}$. If u_{ij} represents exact values of u at the grid points, then the second derivatives u_{xx} and u_{yy} can be written as

$$u_{xx} = \frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{h^2} + O(h^2), \qquad u_{yy} = \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{h^2} + O(h^2),$$

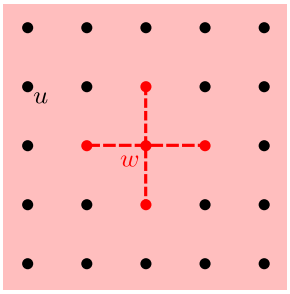

FIGURE 1. A five-point finite difference stencil on a regular grid, shown in red.

which can be verified readily using Taylor expansions. Substituting these approximations into the PDE eigenvalue problem and dropping the $O(h^2)$ terms, we obtain the *discrete* eigenvalue problem

$$\frac{4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} = \tilde{\lambda} u_{ij} \tag{4}$$

for $1 \le i, j \le n-1$, with boundary conditions

$$u_{i,0} = u_{i,n} = u_{0,j} = u_{n,j} = 0.$$

In Figure 1, we show the location of the five unknowns involved in the equation (4) associated with the point w. If we gather all $(n-1)^2$ linear equations and write them in matrix form, we obtain the *matrix eigenvalue problem*

$$A\mathbf{u} = \tilde{\lambda}\mathbf{u},$$

where $\mathbf{u}$ is a vector obtained from the grid function u_{ij} by removing the boundary nodes (whose values are known to be zero), and by imposing a contiguous ordering on the remaining nodes, from 1 to $(n-1)^2$. For example, if we use the lexicographical ordering

$$\mathbf{u}^T = (u_{11}, \ldots, u_{n-1,1} \mid u_{12}, \ldots, u_{n-1,2} \mid \cdots \mid u_{1,n-1}, \ldots, u_{n-1,n-1})^T,$$

then the matrix A has the form

$$A = \begin{bmatrix} T & -I & & & \\ -I & T & -I & & \\ & \ddots & \ddots & \ddots & \\ & & -I & T & -I \\ & & & -I & T \end{bmatrix}, \quad \text{where} \quad T = \begin{bmatrix} 4 & -1 & & & \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{bmatrix}$$

and I is the $(n-1) \times (n-1)$ identity matrix. Note that A is a *sparse matrix* with at most five entries per row, since each equation only contains five unknowns. This motivates the use of eigenvalue solvers for sparse matrices, which we will discuss in Section 4.

2.2. **Discrete Eigenvalues.** For the simple case of a unit square, it is in fact possible to calculate the eigenvalues and eigenvectors analytically. Let $\mathbf{u}$ be the vector corresponding to the grid function (u_{ij}) of the form

$$u_{ij} = \sin(i\theta)\sin(j\phi),$$

where θ and ϕ are fixed constants. Note that this vector satisfies the boundary conditions

$$u_{i,0} = u_{0,j} = 0.$$

Using some elementary trigonometric identities, we can show that

$$4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1} = (4 - 2\cos(\theta) - 2\cos(\phi))u_{ij},$$

so that (4) implies, for $u_{ij} \neq 0$, that

$$\tilde{\lambda} = \frac{4 - 2\cos(\theta) - 2\cos(\phi)}{h^2} = \frac{4}{h^2}\left[\sin^2(\theta/2) + \sin^2(\phi/2)\right].$$

To determine θ and ϕ, we use the remaining boundary conditions

$$\begin{aligned} u_{n,j} = 0 \quad &\Longrightarrow \quad \sin(n\theta) = 0 \\ u_{i,n} = 0 \quad &\Longrightarrow \quad \sin(n\phi) = 0 \end{aligned}$$

Thus, the possible θ and ϕ values are

$$\theta_k = \frac{k\pi}{n}, \quad \phi_\ell = \frac{\ell\pi}{n} \quad \text{for } 1 \leq k, l \leq n-1.$$

It is possible to show that for different (k, ℓ) pairs, the resulting eigenfunctions are mutually orthogonal, so they are linearly independent. The associated eigenvalues are

$$\tilde{\lambda}_{k\ell} = 4n^2\left[\sin^2\Big(\frac{k\pi}{2n}\Big) + \sin^2\Big(\frac{\ell\pi}{2n}\Big)\right]$$

for $k = 1, \ldots, n-1$, $\ell = 1, \ldots, n-1$. Because discrete sine functions of different frequencies are mutually orthogonal, we have found $(n-1)^2$ linearly independent eigenfunctions, so we have found all the eigenvalues. Note that some of them have multiplicities higher than 1, because of the symmetry between k and ℓ.

We now compare with the discrete eigenvalues with those of the **continuous** problem, which we found in Section 1 to be

$$\lambda_{k\ell} = \pi^2(k^2 + \ell^2).$$

Theorem 2.1 (Convergence of discrete eigenvalues)**.** *For the Laplacian problem on the unit square, the discrete eigenvalues $\tilde{\lambda}_{k\ell}$ obtained from a finite difference discretization with mesh size $h = 1/n$, $n > k, l$, satisfy $\tilde{\lambda}_{k\ell} \leq \lambda_{k\ell}$ with*

$$\frac{|\tilde{\lambda}_{k\ell} - \lambda_{k\ell}|}{|\lambda_{k\ell}|} \leq C(k, \ell)h^2,$$

where $C(k, \ell) = \frac{1}{12}\max(k^2, \ell^2)$ is independent of h but dependent on k and ℓ.

Proof. Using the definition $h = 1/n$, we calculate

$$\lambda_{k\ell} - \tilde{\lambda}_{k\ell} = k^2\pi^2\left[1 - \Big(\frac{2}{k\pi h}\sin(k\pi h/2)\Big)^2\right] + \ell^2\pi^2\left[1 - \Big(\frac{2}{\ell\pi h}\sin(\ell\pi h/2)\Big)^2\right]. \tag{5}$$

Noting that for $0 < \theta < \pi/2$, we have

$$1 - \frac{\theta^2}{6} \leq \frac{\sin\theta}{\theta} \leq 1,$$

we immediately deduce that $\tilde{\lambda}_{k\ell} \leq \lambda_{k\ell}$ for any choice of k and ℓ. Moreover, we have

$$1 - \frac{\sin^2\theta}{\theta^2} = \left(1 - \frac{\sin\theta}{\theta}\right)\left(1 + \frac{\sin\theta}{\theta}\right) \leq \frac{\theta^2}{6} \cdot 2 = \frac{\theta^2}{3},$$

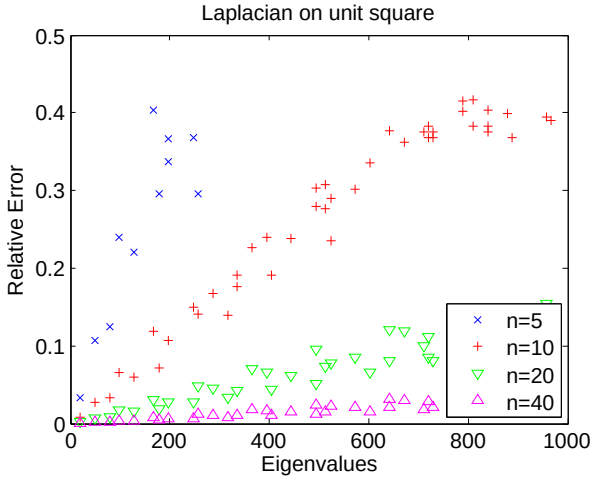

FIGURE 2. Relative error of the eigenvalues on the unit square obtained from a finite difference discretization for different mesh sizes.

so applying this estimate to (5) with $\theta = k\pi h/2$ and $\theta = \ell\pi h/2$ leads to

$$0 \le \lambda_{k\ell} - \tilde{\lambda}_{k\ell} \le \frac{(k\pi)^4 + (\ell\pi)^4}{12} h^2 \le \frac{\max(k^2, \ell^2)\pi^2}{12} \lambda_{k\ell} h^2,$$

from which the conclusion follows. □

To illustrate Theorem 2.1, we plot in Figure 2 the relative error $|\lambda_{k\ell} - \tilde{\lambda}_{k\ell}|/|\lambda_{k\ell}|$ of discrete eigenvalues $\tilde{\lambda}_{k\ell}$ for different values of n. We see that as we increase n, the relative error becomes smaller, meaning they become more and more accurate. Nonetheless, the smallest eigenvalues are much better approximated than the larger ones, especially for small n. Indeed, the relative error is much worse for the larger eigenvalues than for the small ones. This is consistent with the statement of the theorem, where the constant in front of the relative error grows with k and ℓ. Thus, for the larger eigenvalues, a much finer grid is needed to obtain a good approximation.

2.3. **L-shaped Domain.** The real power of numerical methods lies in their ability to produce approximate eigenvalues when they cannot be computed analytically. Consider the L-shaped domain shown in the left panel of Figure 3. We wish to study the eigenvalues of the Laplacian on this domain with Dirichlet boundary conditions; in other words, we consider

$$-\Delta u = \lambda u,$$
$$u|_{\partial\Omega} = 0.$$

The finite difference method with a regular grid (4) can be used here, with the only differences being that there are no unknowns corresponding to points in the second quadrant, and that all nodes along the edges $\{0\} \times [0, 1]$ and $[-1, 0] \times \{0\}$ need

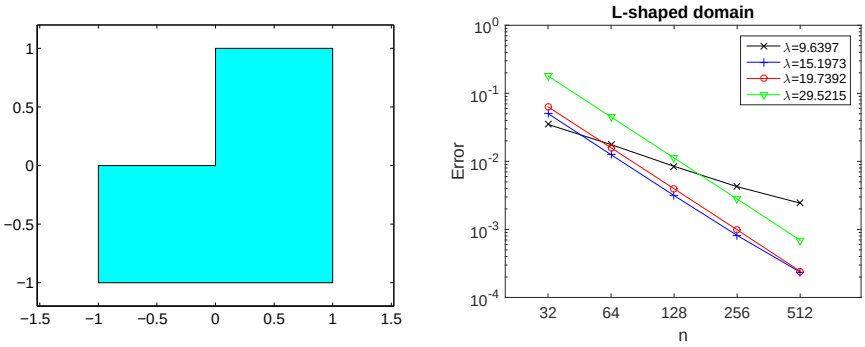

FIGURE 3. Left: an L-shaped domain. Right: Error of the first four eigenvalues as a function of the grid size n.

to take on the value zero. In Matlab, the matrix A can be generated conveniently using the commands

```
G = numgrid('L',n+1);
A = n^2/4*delsq(G);
```

We then calculate the four smallest eigenvalues of the resulting matrix for mesh sizes $h = 1/n$. Since we do not know the exact eigenvalues in the continuous case, we approximate them by *extrapolation*: assuming that each discrete eigenvalue behaves like $\tilde{\lambda}_{k\ell} = \lambda_{k\ell} + Ch^\alpha$, we use the ε-algorithm by P. Wynn (cf. [21, §5.2.4]) to obtain its limit as $h \to 0$. We then subtract this value from the discrete eigenvalues to compute the error, which is plotted on the right panel of Figure 3. Moreover, we plot corresponding eigenfunctions in Figure 4. Note the resemblance between the first eigenfunction and the MATLAB logo. The latter is in fact an approximation of the eigenfunction, obtained using the *Method of Particular Solutions*; see Section 6 for more details and references.

Regarding the approximation errors, we observe that the first eigenvalue behaves differently from the other three: whereas the error curves for the other three have the same slope, and behaves like $O(h^2)$, the first eigenvalue converges more slowly, behaving more like $O(h^\alpha)$ with $1 < \alpha < 2$. In fact, the reason has to do with the regularity of the eigenfunction itself: whereas the other three eigenfunctions appear to be smooth, the first eigenfunction has a fold near the re-entrant corner and is not sufficiently differentiable near it. As a result, the Taylor expansion that was used to derive the finite difference method is no longer valid near the re-entrant corner, and the method suffers from loss of accuracy as a result.

2.4. **More General Boundary Conditions.** Let us now consider the Laplace eigenvalue problem on the unit square $\Omega = (0,1) \times (0,1)$ with *Neumann boundary conditions*

$$-\Delta u = \lambda u, \qquad \nabla u \cdot \mathbf{n} = 0 \text{ on } \partial\Omega.$$

The main difference between this problem and the one with Dirichlet boundary conditions is that the nodes on the boundary no longer assume the value zero, but are instead unknown quantities that need to be solved, just like interior points. One possibility is to use ghost point techniques, such as the one introduced in

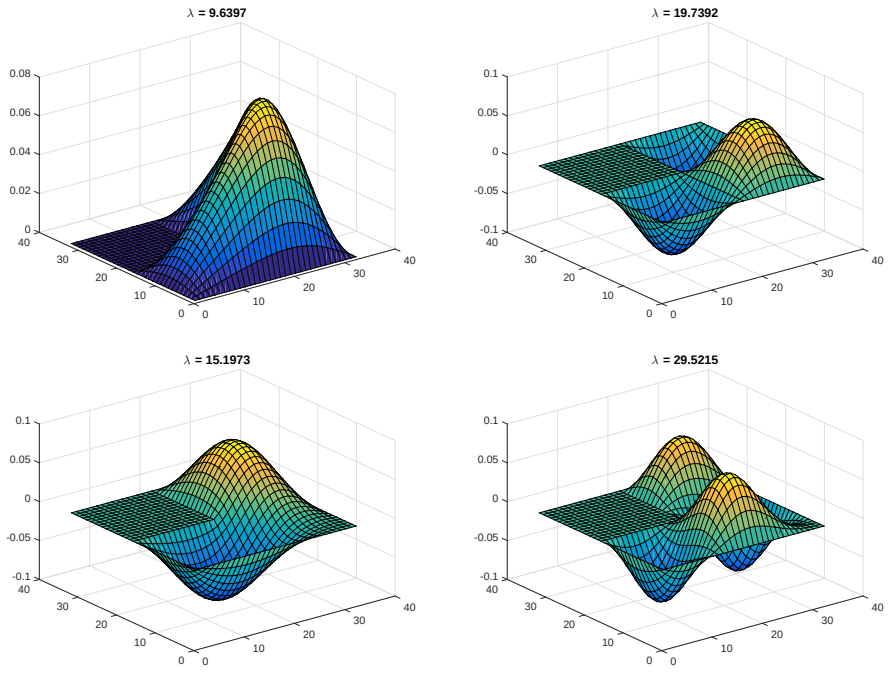

FIGURE 4. Eigenfunctions corresponding to the first four eigenvalues of the L-shaped domain.

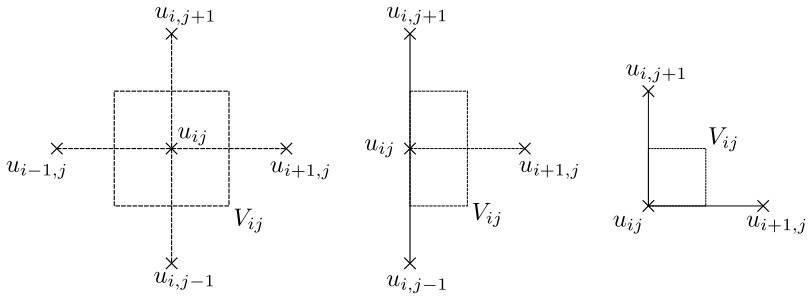

FIGURE 5. Control volume for an interior point (left panel), an edge point (middle panel), and a corner point (right panel).

Section 5.2; however, unless a special scaling is used, this approach leads to a non-symmetric matrix A, which is undesirable for a normal operator, especially from a numerical point of view, see Section 4. Here, we consider a different possibility, known as a *finite volume discretization*, to handle boundary conditions involving normal derivatives.

The idea of a finite volume discretization to simply integrate the equation $-\Delta u = \lambda u$ over a small *control volume* around a grid point u_{ij}. On a regular grid in two

dimensions, the control volume V_{ij} for an interior point is simply an $h \times h$ square cell centered at (x_i, y_j), see the left panel of Figure 5. Integrating the left hand side over V_i and applying the divergence theorem gives

$$-\int_{V_{ij}} \Delta u \, dx = \int_{\partial V_{ij}} \nabla u \cdot \mathbf{n} \, dS(x) \approx h \left[\frac{u_{ij} - u_{i+1,j}}{h} + \frac{u_{ij} - u_{i-1,j}}{h} + \frac{u_{ij} - u_{i,j+1}}{h} + \frac{u_{ij} - u_{i,j-1}}{h} \right],$$

where we approximate the integrals along each edge by the appropriate finite difference, multiplied by the length of the edge. The right hand side is simply approximated by

$$\int_{V_{ij}} \lambda u \, dx \approx \lambda h^2 u_{ij}.$$

Thus, we get the same equation as in the finite difference method, up to a multiplicative factor of h^2.

On the left boundary, we know that the normal derivative is zero, so there is one fewer term to approximate in the divergence theorem. Using the same notation as in the middle panel of Figure 5, we get for the left hand side

$$\int_{\partial V_{ij}} \nabla u \cdot \mathbf{n} \, dS(x) \approx \frac{h}{2} \left[\frac{u_{ij} - u_{i,j+1}}{h} + \frac{u_{ij} - u_{i,j-1}}{h} \right] + h \cdot \frac{u_{ij} - u_{i+1,j}}{h},$$

whereas the right hand side reads

$$\int_{\partial V_{ij}} \lambda u \, dx \approx \frac{h^2}{2} u_{ij},$$

where the factor $1/2$ comes from the fact that the area of V_{ij} is only half of that of the other control volumes. The other edge points can be handled similarly. Finally, for the corner grid point shown on the right panel of Figure 5 , the stencil reads

$$\int_{\partial V_{ij}} \nabla u \cdot \mathbf{n} \, dS(x) \approx \frac{h}{2} \left[\frac{u_{ij} - u_{i+1,j}}{h} + \frac{u_{ij} - u_{i,j+1}}{h} \right],$$

and the right-hand side carries a factor of $1/4$. Collecting all these equations and using the matrix notation, we obtain the *generalized eigenvalue problem*

$$A\mathbf{u} = \tilde{\lambda} B \mathbf{u}, \tag{6}$$

where $B = \mathrm{diag}(b_i)$ is diagonal with entries

$$b_i = \begin{cases} h^2, & \text{at interior points,} \\ h^2/2, & \text{at edge points,} \\ h^2/4, & \text{at corners points.} \end{cases}$$

Note that both A and B are symmetric matrices. Moreover, we can rewrite B as $B = D^2$, where $D = \mathrm{diag}(\sqrt{b_i})$ is a non-singular diagonal matrix. Using the transformation $\mathbf{v} = D\mathbf{u}$, we obtain the equivalent eigenvalue problem

$$D^{-1} A D^{-1} \mathbf{v} = \tilde{\lambda} \mathbf{v}, \tag{7}$$

where the matrix $D^{-1}AD^{-1}$ remains symmetric. Note that although (7) has the same eigenvalues as (6), the eigenvectors are not identical; in fact, the entries of the eigenvectors $\mathbf{v}$ do not approximate the corresponding continuous eigenfunction at the grid points, because of the extra scaling by D. Fortunately, the Matlab

built-in commands `eig` and `eigs` know how to solve generalized as well as standard eigenvalue problems, so users do not need to transform (6) into (7) manually.

3. Finite Element Methods

The finite element method is perhaps one of the most versatile and commonly used discretizations in the numerical solution of partial differential equations. It is based on a different formulation of the PDE, the so-called *weak* or *variational formulation*, which is obtained from integration by parts.

3.1. Variational Formulation.

Consider the following Laplace eigenvalue problem: find a scalar λ and a non-zero function u such that

$$-\Delta u = \lambda u \quad \text{on } \Omega, \qquad u|_{\partial\Omega} = 0. \tag{8}$$

To obtain the variational formulation, let v be another sufficiently smooth function on Ω such that $v|_{\partial\Omega} = 0$. Then multiplying (8) by v and integrating over Ω, we get

$$\begin{aligned} -\int_\Omega v\Delta u &= \lambda \int_\Omega uv \\ -\underbrace{\int_{\partial\Omega} v\nabla u\cdot n}_{=0} + \int_\Omega \nabla u\cdot\nabla v &= \lambda\int_\Omega uv, \end{aligned} \tag{9}$$

where the first term vanishes because v is zero on the boundary. Note that (9) holds for many functions v; in fact, we can define a linear space of functions V such that (9) holds as long as $v \in V$. This function space V would include all v such that

1. $\int_\Omega v^2 < \infty$ (RHS integral defined),
2. $\int_\Omega |\nabla v|^2 < \infty$ (LHS integral defined),
3. $v|_{\partial\Omega} = 0$ (boundary conditions satisfied).

The space of functions satisfying (1) and (2), i.e., functions whose values and partial derivatives are square integrable, is known as the Sobolev space $H^1(\Omega)$, see [18, Chapter 5]; if we require all partial derivatives up to order k to be square integrable, then the resulting space is known as $H^k(\Omega)$. The subspace of $H^1(\Omega)$ of functions that also satisfy (3) is called $H^1_0(\Omega)$, where the subscript zero indicates that functions in this space vanish on the boundary.[2] Thus, the *weak* or *variational form* of the eigenvalue problem is: find $u \in V = H^1_0(\Omega)$ such that

$$a(u,v) = \lambda(u,v) \quad \text{for all } v \in V, \tag{W}$$

where $a(u,v) = \int_\Omega \nabla u\cdot\nabla v$ on the left is a bilinear form with domain $V\times V$, λ is the eigenvalue, and $(\cdot,\cdot)$ on the right denotes the standard L^2 inner product on Ω, i.e., $(u,v) = \int_\Omega uv$. The function space in which we look for solution candidates u is called the *trial space*, and the function space of all v is called the *test space*. In the above weak formulation, the test and trial spaces are the same, although this is not always required.

Variational formulations can also be derived for problems related to (8). The following are a few examples. We leave their derivations as an exercise.

[2] In reality, the definitions of $H^1(\Omega)$ and $H^1_0(\Omega)$ involve weak derivatives and are somewhat more technical, but for the purpose of these notes, the informal definitions given above will suffice.

1. Neumann boundary conditions: for the problem

$$-\Delta u = \lambda u \quad \text{on } \Omega, \qquad \nabla u \cdot n|_{\partial\Omega} = 0,$$

the weak form is: find $u \in H^1(\Omega)$ such that

$$a(u,v) = \lambda(u,v) \qquad \forall v \in H^1(\Omega).$$

Note that the only difference between this and the Dirichlet problem is in the trial and test spaces: for the Neumann problem, it is all of $H^1(\Omega)$, rather than $H^1_0(\Omega)$. Also note that the solution u of this problem automatically satisfies the Neumann boundary condition, and it is not necessary to include a restriction in the trial space. Such conditions are called *natural* boundary conditions. On the other hand, Dirichlet conditions need to be imposed explicitly in the trial and test spaces; these are known as *essential* boundary conditions.

2. Robin boundary conditions: for the problem

$$-\Delta u = \lambda u \quad \text{on } \Omega, \qquad \nabla u \cdot n + pu|_{\partial\Omega} = 0,$$

the weak form is: find $u \in H^1(\Omega)$ such that

$$\tilde{a}(u,v) = \lambda(u,v) \qquad \forall v \in H^1(\Omega)$$

where

$$\tilde{a}(u,v) = \int_\Omega \nabla u \cdot \nabla v + \int_{\partial\Omega} puv.$$

3. Variable diffusivity: for the problem

$$-\nabla \cdot (\kappa(\mathbf{x})\nabla u) = \lambda u \quad \text{on } \Omega, \qquad u|_{\partial\Omega} = 0,$$

the weak form is: find $u \in H^1_0(\Omega)$ such that

$$a_\kappa(u,v) = \lambda(u,v) \qquad \forall v \in H^1_0(\Omega)$$

where

$$a_\kappa(u,v) = \int_\Omega \kappa(\mathbf{x})\nabla u \cdot \nabla v.$$

3.2. **Finite Elements.** To obtain a numerical method for the problem (W), we replace the infinite dimensional space $H^1_0(\Omega)$ by a *finite dimensional* subspace V_h; this is known as a *Ritz–Galerkin Approximation*. Thus, the problem becomes: Find $u_h \in V_h$ such that

$$(\mathrm{W}_h) \qquad a(u_h, v_h) = \lambda \int_\Omega u_h v_h \quad \text{for all } v_h \in V_h,$$

where $a(u,v) = \int_\Omega \nabla u \cdot \nabla v$. The above problem is finite dimensional and can thus be rewritten (and solved) as a matrix eigenvalue problem, which we will show in detail in the next section. Different *finite element methods* correspond to different choices of the subspace V_h; for a general reference on finite element methods, see [27].

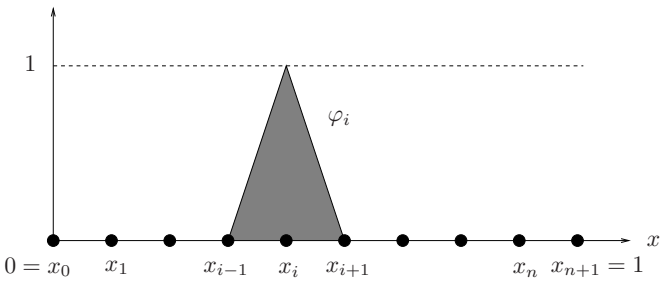

FIGURE 6. A hat function in 1D.

Example 3.1. *Let $\Omega = (0,1) \subset \mathbb{R}$ be equipped with the partition*

$$0 = x_0 < x_1 < \cdots < x_n < x_{n+1} = 1.$$

If we choose V_h to be the set of continuous functions that is linear within each interval $I_j = (x_{j-1}, x_j)$, $j = 1, \ldots, n+1$, then V_h is a finite dimensional space spanned by the "hat functions" $(\varphi_i)_{i=1}^n$, where

$$\varphi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}}, & x \in [x_{i-1}, x_i], \\ \frac{x_{i+1} - x}{x_{i+1} - x_i}, & x \in [x_i, x_{i+1}], \\ 0 & \text{otherwise.} \end{cases}$$

Figure 6 shows a typical hat function. These hat functions are also known as P^1 finite element shape functions, because they are piecewise linear. Note that the degrees of freedom are located at the grid points $x_1, \ldots, x_n$, and the derivative of any $\varphi_h \in V_h$ is a piecewise constant function. Higher order elements, e.g., piecewise quadratic, cubic, etc., are also possible.

Example 3.2. *Suppose $\Omega \subset \mathbb{R}^2$ is a 2D polygonal domain. Then it can be* triangulated, *i.e., decomposed into a union of disjoint triangles, which we denote by $\mathcal{T}_h$, see Figure 7 for an example. Then we can define a set of hat functions similarly to the one-dimensional case: at each node i of the triangulation, we associate the hat function ϕ_i that is linear on each triangle, whose value is 1 at node i and zero at all the other nodes. Not surprisingly, they are also called P^1 finite element shape functions, see Figure 8 for an illustration. These hat functions span V_h, the set of all continuous functions that is piecewise linear on each triangle in $\mathcal{T}_h$. V_h is again a finite dimensional space, and the Ritz–Galerkin problem of finding $u_h \in V_h$ such that*

$$a(u_h, v_h) = \lambda \int_\Omega u_h v_h \qquad \text{for all } v_h \in V_h$$

can again be rewritten as a matrix eigenvalue problem, as we will show below.

3.3. **Matrix Problem.** Let $V_h = \text{Span}\{\varphi_1, \ldots, \varphi_N\}$. Letting $u_h = \sum_j u_j \varphi_j$ and $v_h = \varphi_i$ in the weak form, we see that

$$\sum_j u_j a(\varphi_j, \varphi_i) = \lambda \sum_j u_j \int_\Omega \varphi_i \varphi_j, \qquad i = 1, \ldots, N.$$

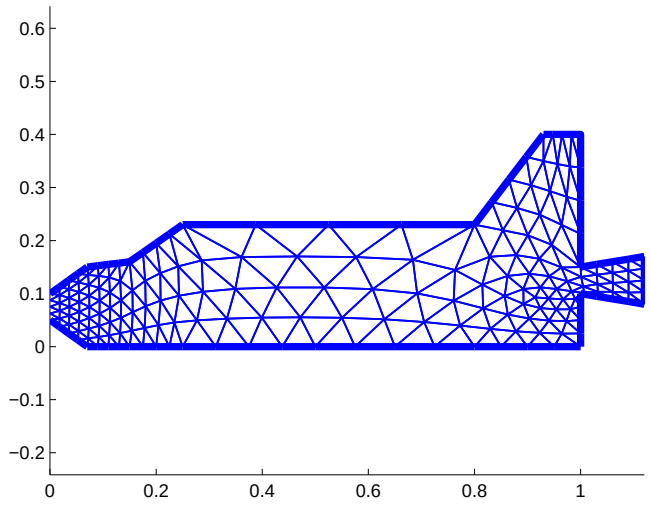

FIGURE 7. Triangulation of a polygonal domain.

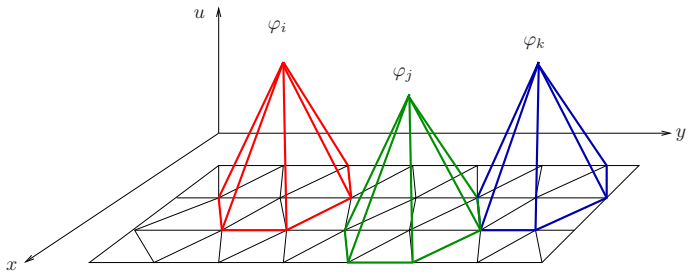

FIGURE 8. Hat functions in 2D.

Rewriting the above equation in matrix form yields the *generalized eigenvalue problem*

$$K\mathbf{u} = \lambda M\mathbf{u},$$

where $K = (K_{ij})_{i,j=1}^N$ and $M = (M_{ij})_{i,j=1}^N$ with

$$K_{ij} = a(\varphi_i, \varphi_j) = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j, \quad M_{ij} = \int_\Omega \phi_i\phi_j.$$

K is often called the *stiffness* matrix, and M the *mass* matrix. These matrices enjoy the following properties:

- M and K are symmetric;
- M is positive definite;
- K is positive definite for the Dirichlet and Robin problems, and is positive semi-definite for the Neumann problem.

Moreover, M and K are *sparse*. Recall the definition of the (i,j)-th entries of K and M:

$$K_{ij} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j, \quad M_{ij} = \int_\Omega \varphi_i\varphi_j.$$

If nodes i and j do not share a common triangle (or interval in 1D), then the support of φ_i and φ_j are disjoint. In this case, the corresponding integrals must vanish, so $K_{ij} = M_{ij} = 0$ there. In the 1D case, this means a degree of freedom can only be coupled to itself and its neighbours, so the matrices K and M are *tridiagonal* if the degrees of freedom are numbered contiguously from left to right.

Example 3.3. *For the Dirichlet problem in 1D, hat functions with uniform mesh size h yields*

$$M_{ij} = \begin{cases} 0, & |i-j| > 1, \\ \int_0^h h^{-2}x(h-x)\,dx = h/6, & |i-j| = 1, \\ 2\int_0^h h^{-2}x^2\,dx = 2h/3, & i = j. \end{cases}$$

In 2D, the matrices K and M can be *assembled* from individual contributions from each triangle. To do so, note that the integrals in the definition of K_{ij} and M_{ij} can be written as

$$K_{ij} = \sum_{T\in\mathcal{T}_h} \int_T \nabla\varphi_i \cdot \nabla\varphi_j, \quad M_{ij} = \sum_{T\in\mathcal{T}_h} \int_T \varphi_i\varphi_j.$$

Thus, if integrals of the form $\int_T \varphi_i\varphi_j$ and $\int_T \nabla\varphi_i \cdot \nabla\varphi_j$ are known for all triangles T, then it suffices to put these quantities in the right positions in the matrices K and M and add up all the contributions. On the other hand, for a fixed triangle T, the only non-zero integrals are those for which i and j are vertices of T; all the other integrals vanish, because T would be outside the support of φ_i or φ_j in that case. Thus, at most $12|\mathcal{T}_h|$ integrals are needed in order to compute K and M, where $|\mathcal{T}_h|$ is the number of triangles in the triangulation. This leads to the following *assembly process* for calculating K and M:

1. Define the mesh and number the nodes.
2. Calculate the *element* stiffness and mass matrices
$$[K_i]_{jk} = \int_{\mathcal{T}_i} \nabla\varphi_j \cdot \nabla\varphi_k.$$
3. Add the element matrices into the right places in the assembled matrices, i.e., if
$$K_i := \begin{bmatrix} (\nabla\varphi_j,\nabla\varphi_j)_{T_i} & (\nabla\varphi_j,\nabla\varphi_k)_{T_i} & (\nabla\varphi_j,\nabla\varphi_l)_{T_i} \\ & (\nabla\varphi_k,\nabla\varphi_k)_{T_i} & (\nabla\varphi_k,\nabla\varphi_l)_{T_i} \\ \text{Sym.} & & (\nabla\varphi_l,\nabla\varphi_l)_{T_i} \end{bmatrix} =: \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ & p_{22} & p_{23} \\ \text{Sym.} & & p_{33} \end{bmatrix},$$
update the global stiffness matrix K by
$$K \leftarrow K + \begin{bmatrix} 0 & & & & & 0 \\ & p_{11} & & p_{12} & & p_{13} \\ & & 0 & & 0 & \\ & p_{12} & & p_{22} & & p_{23} \\ & & 0 & & 0 & \\ & p_{13} & & p_{23} & & p_{33} \\ 0 & & & & & 0 \end{bmatrix} \begin{matrix} \\ j \\ \\ k \\ \\ l \\ \\ \end{matrix},$$
$$\begin{matrix} & j & & k & & l & \end{matrix}$$

and similarly for M.

4. Repeat for all the other triangles.
5. Remove rows and columns associated with Dirichlet boundary nodes.

A sample finite element code is provided at `http://www.math.hkbu.edu.hk/~felix_kwok/crm/`. More information can be found in the Section 7.

3.4. **Minimax Principle.** A powerful characterization of eigenvalues, in terms of extrema of certain expressions over subspaces, is known as the *minimax principle*. Let $\mathscr{L}$ be a self-adjoint, positive definite operator in a Hilbert space H, equipped with the inner product $(\cdot,\cdot)$, and let $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_k \le \cdots$ be the eigenvalues of $\mathscr{L}$.

Theorem 3.1 (Minimax principle)**.** *The kth eigenvalue λ_k of $\mathscr{L}$ satisfies*

$$(*) \qquad \lambda_k = \min_{\dim(U)=k} \max_{\substack{u\in U\\ u\neq 0}} R(u),$$

where $R(u) := \dfrac{(u,\mathscr{L}u)}{\|u\|^2}$ *is called the* Rayleigh quotient.

Proof. Let $u_1, u_2, \ldots$ denote the eigenvectors of $\mathscr{L}$ (for existence, see [31]). If $U = \mathrm{Span}(u_1,\ldots,u_k)$, then

$$\max_{\substack{u\in U\\ u\neq 0}} R(u) = \frac{(u_k,\mathscr{L}u_k)}{\|u_k\|^2} = \frac{(u_k,\lambda_k u_k)}{\|u_k\|^2} = \lambda_k.$$

To show that this is the minimum over all choices of k dimensional subspace, let us consider another $U = \mathrm{Span}(w_1,\ldots,w_k)$, this time an arbitrary k-dimensional subspace. We choose $w = \alpha_1 w_1 + \cdots + \alpha_k w_k$ such that $w \neq 0$ and

$$(w,u_1) = \cdots = (w,u_{k-1}) = 0.$$

This is possible because this is in fact a system of $k-1$ equations in k unknowns, so a non-trivial solution always exists. Now since $u_1, u_2, \ldots$ is a basis of H, we can write

$$w = \sum_{i=k}^{\infty} \beta_i u_i,$$

where we used the fact that $\beta_1 = \cdots = \beta_{k-1} = 0$ because of the constraints. Then

$$(w,\mathscr{L}w) = \sum_{i=k}^{\infty} \beta_i^2 (u_i,\mathscr{L}u_i) \ge \lambda_k \sum_{i=k}^{\infty} \beta_i^2 = \lambda_k \|w\|^2.$$

Thus, $R(w) \ge \lambda_k$, so

$$\min_{\dim(U)=k} \max_{\substack{u\in U\\ u\neq 0}} R(u) = \lambda_k.$$

□

Corollary 3.2. *If $V_h \subset H_0^1(\Omega)$ is a finite element subspace, λ_k and $\tilde{\lambda}_k$ are the k-th smallest eigenvalues of the problems (W) and (W_h) respectively, then $\lambda_k \le \tilde{\lambda}_k$.*

Proof. Let $\tilde{u}_1,\ldots,\tilde{u}_k \in V_h \subset H_0^1(\Omega)$ be the eigenfunctions corresponding to the k smallest eigenvalues of the problem (W_h). Then $U = \mathrm{Span}\{\tilde{u}_1,\ldots,\tilde{u}_k\}$ is a k-dimensional subspace of $H_0^1(\Omega)$ with $\max_{u\in U\setminus\{0\}} R(u) = \tilde{\lambda}_k$. Thus, by the minimax principle, we have $\lambda_k \le \tilde{\lambda}_k$, as required. □

TABLE 1. Convergence of the first two eigenvalues of the Laplacian on the unit square, discretized using a P^1 finite element method. The ratios between errors in successive rows indicate $O(h^2)$ convergence.

h	$\tilde{\lambda}_1$	$\tilde{\lambda}_1 - \lambda_1$	Ratio	$\tilde{\lambda}_2$	$\tilde{\lambda}_2 - \lambda_2$	Ratio
1	22.8658	3.1266		62.5602	13.2122	
1/2	20.5055	0.7663	0.2451	52.6298	3.2818	0.2484
1/4	19.9298	0.1906	0.2487	50.1664	0.8184	0.2494
1/8	19.7868	0.0476	0.2497	49.5525	0.2045	0.2499
1/16	19.7511	0.0119	0.2499	49.3991	0.0511	0.2500

Example 3.4. *Consider the first two eigenvalues of unit square, $\lambda_1 = 2\pi^2$, $\lambda_2 = 5\pi^2$. In Table 1, we discretize the square using a finite element method with a regular triangulation. We successively refine the mesh by a factor of 2, so that the space V_h used in each row is a subspace of the V_h in the subsequent rows, and they are all subspaces of $H_0^1(\Omega)$. We see that all the $\tilde{\lambda}_i$ are larger than λ_i, and the $\tilde{\lambda}_i$ decrease as the mesh is refined. We also observe that the error behaves like $O(h^2)$, just like in the finite difference case. This will be proved in the next section.*

Remark 3.1. *Corollary 3.2 shows that the finite element method always produces* over-estimations *of the exact eigenvalues. This is in contrast with finite difference methods in Chapter 1, which can produce approximations that are smaller than the exact eigenvalues, e.g. for the unit square, cf. Theorem 2.1. This is because finite difference approximations cannot be interpreted as the exact Laplacian applied to a subspace of the continuous functions, so the minimax principle does not apply.*

3.5. **Convergence of the Finite Element Method.** Let Ω be a bounded open subset of $\mathbb{R}^2$ and consider the Dirichlet eigenvalue problem

$$-\Delta u = \lambda u \quad \text{on } \Omega, \qquad u|_{\partial\Omega} = 0.$$

Let $0 < \lambda_1 \leq \lambda_2 \leq \cdots$ be the exact eigenvalues, and $0 < \tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq \cdots$ be the eigenvalues of the finite element approximation over the conforming, quasi-uniform mesh $\mathcal{T}_h$.[3] We define the mesh parameter h by

$$h = \max_{T \in \mathcal{T}_h} \operatorname{diam}(T),$$

i.e., h is the maximum diameter of all the triangles in $\mathcal{T}_h$. Our goal in this section is to prove the following theorem:

Theorem 3.3. *Let $\Omega \subset \mathbb{R}^2$ be a convex polygon, $\mathcal{T}_h$ be a conforming, quasi-uniform triangulation of Ω, and V_h be the P^1 finite element space associated with $\mathcal{T}_h$. If $\tilde{u}_k$ is an eigenfunction corresponding to the k-th smallest eigenvalue $\tilde{\lambda}_k$ of the discrete problem*

$$a(\tilde{u}_k, v_h) = \tilde{\lambda}_k(\tilde{u}_k, v_h) \quad \forall v_h \in V_h,$$

then for $h > 0$ small enough, we have $|\tilde{\lambda}_k - \lambda_k| \leq C(k)h^2$, where $C(k)$ is a constant depending on k and Ω but independent of h. In particular, $\tilde{\lambda}_k$ converges to the exact eigenvalue λ_k as $h \to 0$.

[3]For a precise definition, see [13].

To do so, we need to understand how well the finite element space V_h approximates the invariant subspaces of $-\Delta$, which are spanned by the eigenfunctions $u_1, u_2, \ldots$. This leads us to consider the following projection operator.

Definition 3.1. *The operator $P_h : H_0^1(\Omega) \to V_h$ defined by*

$$a(P_h u, v_h) = a(u, v_h) \quad \forall v_h \in V_h.$$

This operator is well defined because the stiffness matrix K is symmetric positive definite. The following properties of P_h can be readily verified:

(i) P_h is a *projection* onto V_h,
(ii) $a(u - P_h u, v_h) = 0$ for all $v_h \in V_h$,
(iii) $a(P_h u, P_h u) \le a(u, u)$ for all $u \in H_0^1(\Omega)$.

The next step is to estimate the norm of $\|u - P_h u\|_{L^2(\Omega)}$. This will require the following tools:

1. A best approximation estimate,
2. An interpolation estimate,
3. A duality argument.

1. Best Approximation Estimate. The best approximation result is due to the following lemma.

Theorem 3.4 (Céa's Lemma)**.** *For all $w_h \in V_h$, we have*

$$a(u - P_h u, u - P_h u) \le a(u - w_h, u - w_h).$$

Proof. For any $w_h \in V_h$, we have

$$\begin{aligned} a(u - w_h, u - w_h) &= a(u - P_h u + \underbrace{(P_h u - w_h)}_{\in V_h}, u - P_h u + (P_h u - w_h)) \\ &= a(u - P_h u, u - P_h u) + \underbrace{a(P_h u - w_h, P_h u - w_h)}_{\ge 0} \\ &\ge a(u - P_h u, u - P_h u). \end{aligned}$$

□

Since $a(u - P_h u, u - P_h u) = \|\nabla(u - P_h u)\|^2_{L^2(\Omega)}$, Céa's lemma says that the elliptic projector chooses from the subspace V_h the function u_h that minimizes the gradient of the error in the L^2 sense. On the other hand, the error can still be large if u is very far away from V_h. Thus, we need to quantify how well the function u is approximated by functions in V_h. This is known as an *interpolation estimate*.

2. Interpolation Estimate. If u is a continuous function, then we define u_I to be its piecewise polynomial interpolant of degree r on each element:

$$u_I = \sum_j u(\mathbf{x}_j)\varphi_j,$$

where $\mathbf{x}_j$ denotes the j-th node in the grid. Note carefully that u_I is in general different from the $P_h u$ obtained from the elliptic projector: although both functions are in V_h, $P_h u$ is defined by a minimization problem and need not interpolate the continuous function exactly, whereas u_I is an interpolant that need not minimize anything. For functions in $H_0^1(\Omega)$ that are discontinuous, there are ways of defining a quasi-interpolant based on averaged Taylor polynomials. For details, see [13].

With the above definitions, we have the following general approximation result.

Theorem 3.5 (Interpolation Error, cf. [41, Lemma B.6])**.** *Let $\mathcal{T}_h$ be a conforming, shape-regular triangulation of $\Omega \subset \mathbb{R}^d$ with maximal element diameter h. Let $0 \leq m \leq s$ and $d/2 < s \leq r+1$, where r is the degree of of the piecewise polynomial on each element. Then there exists a constant C depending only on m, s and the aspect ratio (i.e., the ratio of radii between the circumscribed and inscribed circles) of the elements in $\mathcal{T}_h$, such that*

$$\|D^m(u - u_I)\|_{L^2(\Omega)} \leq Ch^{s-m}\|D^s u\|_{L^2(\Omega)},$$

where D^k denotes the vector of all k-th order (weak) partial derivatives.

The proof of the above theorem in its full generality is rather technical and can be found in [13, Theorem 4.4.4]. Nonetheless, we are able to show the result for the simple one-dimensional case $\Omega = (0,1)$ and piecewise linear interpolation as follows. In an interval $K = [x_{i-1}, x_i]$ of length h, we have

$$\begin{aligned}\|u' - u_I'\|_{L^2(K)}^2 &= \int_{x_{i-1}}^{x_i} (u' - u_I')^2 \, d\xi \\ &= -\int_{x_{i-1}}^{x_i} (u - u_I)(u'' - u_I'') \, d\xi \leq \|u - u_I\|_{L^2(K)} \, \|u''\|_{L^2(K)},\end{aligned}$$

where the boundary terms vanish because of the interpolation property $u(x_i) = u_I(x_i)$. On the other hand, since for all $x \in K$ we have

$$|u(x) - u_I(x)|^2 = \left| \int_{x_{i-1}}^{x} (u' - u_I') \, d\xi \right|^2 \leq (x - x_{i-1})\|u' - u_I'\|_{L^2(K)}^2,$$

we can estimate $\|u - u_I\|_{L^2(K)}$ by

$$\|u - u_I\|_{L^2(K)}^2 \leq \int_{x_{i-1}}^{x_i} (\xi - x_{i-1})\|u' - u_I'\|_{L^2(K)}^2 \, d\xi = \frac{h^2}{2}\|u' - u_I'\|_{L^2(K)}^2.$$

Thus, summing over all intervals gives

$$\|u' - u_I'\|_{L^2(\Omega)} \leq \frac{h}{\sqrt{2}}\|u''\|_{L^2(\Omega)}, \qquad \|u - u_I\|_{L^2(\Omega)} \leq \frac{h^2}{2}\|u''\|_{L^2(\Omega)},$$

which is just a special case of Theorem 3.5 for $d = 1$, $s = 2$, $r = 1$ and $m = 0, 1$.

3. Duality Argument. We are now ready to estimate $\|u - P_h u\|_{L^2(\Omega)}$ when Ω is a convex polygon. We know from elliptic regularity theory [18, Chapter 6] that if $f \in L^2(\Omega)$ and $a(u, v) = (f, v)$ for all $v \in H_0^1(\Omega)$, then $u \in H^2(\Omega)$, and

$$\|D^2 u\|_{L^2(\Omega)} \leq C\|f\|_{L^2(\Omega)}.$$

Thus, if $u \in H_0^1(\Omega)$ is an eigenfunction with eigenvalue λ, then letting $f = \lambda u$ shows that in fact $u \in H^2(\Omega)$. Using the best approximation property and then the interpolation estimate, we get

$$a(u - P_h u, u - P_h u) \leq \|\nabla(u - u_I)\|_{L^2(\Omega)}^2 \leq Ch^2\|D^2 u\|_{L^2(\Omega)}^2 \leq C'h^2\|f\|_{L^2(\Omega)}^2.$$

To estimate $\|u - P_h u\|_{L^2}$, we use a duality argument known as the *Aubin–Nitsche* trick: consider the auxiliary problem

$$-\Delta\psi = u - P_h u, \qquad \psi|_{\partial\Omega} = 0, \tag{10}$$

or, in weak form,

$$a(\psi, v) = (u - P_h u, v) \quad \forall v \in H_0^1(\Omega).$$

Choosing $v = u - P_h u$ in the weak form yields

$$\begin{aligned}(u - P_h u, u - P_h u) = a(u - P_h u, \psi) &= a(u - P_h u, \psi - P_h \psi) \\ &\leq a(u - P_h u, u - P_h u)^{1/2} a(\psi - P_h \psi, \psi - P_h \psi)^{1/2} \\ &\leq \sqrt{CC'} h^2 \|u - P_h u\|_{L^2(\Omega)} \|D^2 u\|_{L^2(\Omega)},\end{aligned}$$

where we used the fact that ψ satisfies (10) to deduce that $a(\psi - P_h\psi, \psi - P_h\psi)^{1/2} \leq \sqrt{C'} h \|u - P_h u\|_{L^2(\Omega)}$. Thus, we have the estimate

$$\|u - P_h u\|_{L^2(\Omega)} \leq \sqrt{CC'} h^2 \|D^2 u\|_{L^2(\Omega)},$$

provided that Ω is convex, which is needed to guarantee that $D^2 u$ is in $L^2(\Omega)$. We are finally ready to prove Theorem 3.3.

Proof of Theorem 3.3. Consider the k-dimensional subspaces

$$E_k = \mathrm{Span}\{u_1, \ldots, u_k\}, \quad \tilde{E}_k = P_h E_k,$$

where u_k is the eigenfunction of the continuous operator corresponding to the k-th smallest eigenvalue λ_k. For h small enough, we have for all non-zero $v \in E_k$,

$$\|P_h v\| \geq \|v\| - \|v - P_h v\| \geq (1 - C(k)h^2)\|v\| > 0.$$

Thus, P_h is an isomorphism between E_k and $\tilde{E}_k$, so both subspaces have dimension k. By the minimax principle, we have

$$\begin{aligned}\lambda_k \leq \tilde{\lambda}_k &\leq \max_{\substack{v_h \in \tilde{E}_k \\ v_h \neq 0}} \frac{a(v_h, v_h)}{\|v_h\|^2} = \max_{\substack{v \in E_k \\ v \neq 0}} \frac{a(P_h v, P_h v)}{\|P_h v\|^2} \\ &\leq \max_{\substack{v \in E_k \\ v \neq 0}} \frac{a(v,v)}{\|P_h v\|^2} \leq \underbrace{\max_{\substack{v \in E_k \\ v \neq 0}} \frac{a(v,v)}{\|v\|^2}}_{=\lambda_k} \cdot \max_{\substack{v \in E_k \\ v \neq 0}} \frac{\|v\|^2}{\|P_h v\|^2}.\end{aligned}$$

But since $\|P_h v\| \geq (1 - C(k)h^2)\|v\|$, we conclude that

$$\lambda_k \leq \tilde{\lambda}_k \leq \lambda_k(1 + 2C(k)h^2) + O(h^4),$$

which implies $|\tilde{\lambda}_k - \lambda_k| \leq \tilde{C}(k)h^2 \to 0$ as $h \to 0$. □

Remarks:

1. If the eigenfunction u is not smooth enough, i.e., if it does not belong to $H^2(\Omega)$, like in the case for the L-shaped domain, we still have in general [35]

   $$\tilde{\lambda}_k \leq \lambda_k \Big(1 + C(k) \sup_{\substack{v \in E_k \\ \|v\|=1}} \|v - P_h v\|^2_{H^1(\Omega)}\Big).$$

   However, because of the lack of regularity, the contraction rates will in general be of the form $O(h^\alpha)$ with $\alpha < 2$, i.e., convergence will be worse than in the regular case. In order to recover higher order convergence, various methods can be used, such as adaptive grid refinement, see [17] and the references therein.
2. For the convergence of eigenvectors, see [11].

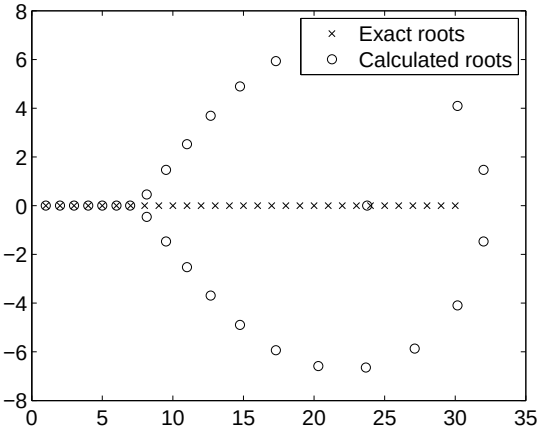

FIGURE 9. The exact and numerically calculated roots of a polynomial of degree 30.

4. Solution of Matrix Eigenvalue Problems

We have seen in Sections 2 and 3 that the we can approximate PDE eigenvalue problems by algebraic eigenvalue problems by discretizing the differential operator in various ways. It remains to solve the algebraic eigenvalue problems to obtain the actual eigenvalues and eigenvectors. In theory, the eigenvalues of a matrix A are given by the roots of its characteristic polynomial $p_A(\lambda) = \det(A - \lambda I)$. It would be tempting to calculate this polynomial explicitly and find its roots using a numerical method.[4] Unfortunately, this is a terrible idea because the roots of $p_A(\lambda)$ are extremely sensitive to small perturbations in the coefficients, as one can see in the example below.

Example 4.1. *Consider the polynomial*[5] $p(\lambda) = (\lambda - 1)(\lambda - 2)\cdots(\lambda - 30)$. *We compute the coefficients of this polynomial in Matlab, and then ask for the roots of this polynomial using the following commands:*

```
A = diag((1:30),0);          % Matrix with 1,2,...,30 on the diagonal
p = poly(A);                 % Characteristic polynomial of A
lambda = roots(p);           % Roots of p
```

We plot the roots `lambda` *thus obtained in Figure 9. We see that the numerically calculated roots are very far from the exact ones, even though Matlab computes with 16 digits of accuracy. The reason is that the largest coefficient in* `p` *is approximately* 2×10^{33}*, so even a perturbation in the 16th significant digit can introduce a huge error in the calculated roots, as illustrated by the figure.*

Thus, numerical algorithms for finding matrix eigenvalues never form the characteristic polynomial, but instead attempt to transform the matrix directly into

[4]From Galois theory, we know there is no explicit formula for the roots when A is a 5×5 matrix or larger! Fortunately, efficient numerical methods exist, such as `roots` in Matlab, which relies on calculating the eigenvalues of the companion matrix. See also [4] for a recent improvement.

[5]This is adapted from Wilkinson's example in [44], where we have changed the degree from 20 to 30 to make it work in double-precision arithmetic.

diagonal form using a sequence of similarity transformations. There are different methods that are commonly used, depending on the properties of the matrix A and how many eigenvalues we seek:

- The *QR iteration* is generally used when we seek *all* eigenvalues of a *small* and/or *dense* matrix:
- If we seek only a *few* eigenvalues of a dense or sparse matrix, commonly used methods include the power method, shift-and-invert, bisection, etc.
- For large *sparse* matrices from which only a few eigenvalues are sought, methods such as Lanczos and Jacobi-Davidson may be more efficient.

In Matlab, there are two different functions for finding eigenvalues and eigenvectors of a matrix: one uses `eig` to find all eigenvalues of a dense matrix, whereas one uses `eigs` to find a few eigenvectors of a sparse matrix. Before discussing a few of the aforementioned algorithms, we give a quick reminder of the properties of matrix eigenvalue problems:

- In an eigenvalue problem, one seeks $\mathbf{v} \neq 0$ such that $A\mathbf{v} = \lambda\mathbf{v}$, where $A = n \times n$ matrix.
- If A is *diagonalizable*, this is equivalent to finding a *diagonal* matrix Λ and a *non-singular* V such that
$$AV = V\Lambda \iff A = V\Lambda V^{-1}.$$
- For *symmetric* A, V can be taken to be *orthogonal*, i.e., $V = Q$, $Q^TQ = QQ^T = I$.

A related problem is the *generalized* eigenvalue problem: solve $A\mathbf{v} = \lambda B\mathbf{v}$, $A, B = n \times n$ matrices. If B is symmetric and positive definite, then this is equivalent to
$$G^{-T}AG^{-1}\mathbf{z} = \lambda\mathbf{z},$$
where $B = G^TG$ is the Cholesky (or any other related) factorization, and $\mathbf{z} = G\mathbf{v}$. Thus, method for finding the eigenvalues of symmetric A can be used on $G^{-T}AG^{-1}$, but there are often more efficient and numerically stable methods that do not require computing $G^{-T}AG^{-1}$ explicitly. In these notes, we will concentrate on the case of symmetric matrices. For the non-symmetric case, see [23].

4.1. **Power Iterations.** Possibly the simplest method for calculating an eigenvalue of a matrix A is *the power method*. Given an initial vector $\mathbf{x}^{(0)}$, the method produces an estimate $\lambda^{(k)}$ and a new vector $\mathbf{x}^{(k)}$ for $k = 1, 2, 3, \ldots$:

1. $\mathbf{x}^{(k)} = A\mathbf{x}^{(k-1)}$
2. $\lambda^{(k)} = \dfrac{(\mathbf{x}^{(k)})^T A\mathbf{x}^{(k)}}{\|\mathbf{x}^{(k)}\|_2^2}$

In finite precision arithmetic, the first step is usually replaced by
$$\mathbf{x}^{(k)} = \frac{A\mathbf{x}^{(k-1)}}{\|A\mathbf{x}^{(k-1)}\|_2}$$
in order to avoid overflow. If $\mathbf{x}^{(k)}$, after suitable normalization, is a fixed point of the iteration, then it is clearly an eigenvector of A with eigenvalue $\lambda^{(k)}$. Even when $\mathbf{x}^{(k)}$ is not a fixed point, this definition of $\lambda^{(k)}$ minimizes $\|A\mathbf{x}^{(k)} - \lambda\mathbf{x}^{(k)}\|_2$ over all values of λ, so the formula returns the best possible eigenvalue approximation in the L^2 sense.

To understand the behavior of the method, suppose A is diagonalizable with eigenvectors $\mathbf{v}_i$, ordered such that $|\lambda_n| > |\lambda_{n-1}| \geq \cdots \geq |\lambda_1|$, where $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Then if $\mathbf{x}^{(0)} = \sum_i \alpha_i \mathbf{v}_i$, the power iteration generates

$$\mathbf{x}^{(k)} = \frac{\sum_i \alpha_i \lambda_i^k \mathbf{v}_i}{\|\sum_i \alpha_i \lambda_i^k \mathbf{v}_i\|_2} = \alpha_n \mathbf{v}_n + O\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|\right)^k,$$

so $\lambda^{(k)}$ converges to the largest eigenvalue in magnitude λ_n provided $\alpha_n \neq 0$, with the estimate

$$\lambda^{(k)} = \lambda_n + O\left(\left|\frac{\lambda_{n-1}}{\lambda_n}\right|\right)^k.$$

If the matrix A is symmetric, the exponent can be improved to $2k$ from k using the fact that the eigenvectors form an orthonormal basis.

The power method admits a few variants that can be quite useful for calculating a single eigenvalue. If one would like calculate the smallest rather than the largest eigenvalue in magnitude, then one would use the *inverse power method*

$$\mathbf{x}^{(k)} = A^{-1}\mathbf{x}^{(k-1)},$$

with the approximated eigenvalue $\mathbf{x}^{(k)}$ calculated the same way as in the power method. Similarly, if we wish to find the eigenvalue closest to a given value μ, then one would use the *shifted inverse iteration*, or the *shift-and-invert method*

$$\mathbf{x}^{(k)} = (A - \mu I)^{-1}\mathbf{x}^{(k-1)}.$$

Of course, one does not calculate the matrix $(A - \mu I)^{-1}$ explicitly in practice; one would instead pre-calculate the LU factorization of $A - \mu I$ and solve at each iteration the linear system

$$(A - \mu I)\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)}$$

using this factorization. This presents significant savings in terms of computational cost, especially when A is a sparse matrix.

Example 4.2. *Let*

$$A = \begin{bmatrix} 4 & 1 & & \\ 1 & 3 & 1 & \\ & 1 & 2 & 1 \\ & & 1 & 1 \end{bmatrix},$$

whose eigenvalues are

$$\frac{1}{2}\left(5 + \sqrt{11 \pm 2\sqrt{21}}\right) \approx \{0.25471876, 1.8227171, 3.1772829, 4.7452812\}.$$

We apply the power, inverse power and shift-and-invert methods to A, with $\mu = 3$ for the last method. Then we know that the three methods converge to the eigenvalues $\lambda_4 \approx 4.7453$, $\lambda_1 \approx 0.2548$ and $\lambda_3 \approx 3.1773$ respectively. Figure 10 shows the convergence of the different methods. We see that the error of the power method decreases from about 2×10^{-1} at iteration 1 to about 2×10^{-9} at iteration 24, so its contraction factor is approximately

$$\left(\frac{2 \times 10^{-9}}{2 \times 10^{-1}}\right)^{1/23} = 0.4489 \approx 0.4484 = \left(\frac{3.1773}{4.7453}\right)^2.$$

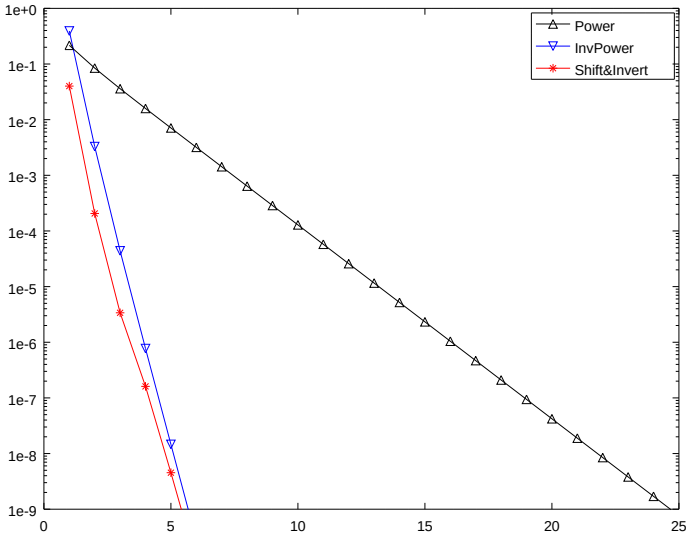

FIGURE 10. Convergence of the power method and its variants for the matrix in Example 4.2.

This nicely matches the stated convergence estimate. Similarly, for the inverse power method and the shift-and-invert method, we obtain as contraction factors

$$\textit{InvPower:} \quad \left(\frac{1.5\times 10^{-8}}{4\times 10^{-1}}\right)^{1/4} = 0.0139 \approx 0.0195 = \left(\frac{0.2548}{1.8227}\right)^{2},$$

$$\textit{ShiftInvert:} \quad \left(\frac{5\times 10^{-9}}{4\times 10^{-2}}\right)^{1/4} = 0.0188 \approx 0.0227 = \left|\frac{\mu - 3.1773}{\mu - 1.8227}\right|^{2}.$$

Note that the ratios are squared in all cases because A is symmetric.

4.2. **Orthogonal iterations and the QR method.** If one wishes to calculate several eigenvalues and eigenvectors at a time, one needs to modify the power method to start with several initial vectors. However, in order to prevent those vectors from converging to the same eigenvector, we need to ensure that they remain linearly independent, which is accomplished by *orthogonalization*. This leads to the so-called *orthogonal iteration*: in the case of two starting vectors, we define the $n\times 2$ matrix $U_0 = [\mathbf{u}_1^{(0)}, \mathbf{u}_2^{(0)}]$ and perform the following iteration for $k = 0, 1, 2, \ldots$:

1. Calculate $Z_k = AU_k$;
2. Factor Z_k into $Z_k = U_{k+1}Y_{k+1}$, where U_{k+1} has orthonormal columns and Y_{k+1} is upper triangular (QR decomposition).

The factorization in step 2 can be computed e.g. by the Gram-Schmidt process. The eigenvalues are approximated by the diagonal entries of

$$T_k = U_k^T A U_k = \begin{bmatrix} (\mathbf{u}_1^{(k)})^T A\mathbf{u}_1^{(k)} & (\mathbf{u}_1^{(k)})^T A\mathbf{u}_2^{(k)} \\ (\mathbf{u}_2^{(k)})^T A\mathbf{u}_1^{(k)} & (\mathbf{u}_2^{(k)})^T A\mathbf{u}_2^{(k)} \end{bmatrix}.$$

Note that the first column $\mathbf{u}_1^{(k)}$ of U_k is identical to the normalized vector $\mathbf{x}^{(k)}$ in the power method, so $(\mathbf{u}_1^{(k)})^T A\mathbf{u}_1^{(k)}$ converges to the largest eigenvalue, assuming that $\mathbf{u}_1$ has a non-zero component along the corresponding eigenvector. The second column, however, cannot converge to same eigenvector, since it is always kept orthogonal from the first column. In fact, $(\mathbf{u}_2^{(k)})^T A\mathbf{u}_2^{(k)}$ converges to the second largest eigenvalue, under some mild assumptions on U_0 and A. Moreover, the off-diagonal entries of T_k can be shown to converge to zero. The method can be extended to any number of starting vectors, as long as they are linearly independent; in this case, it can be shown under certain assumptions (see [2, Theorem 10.6.1] for the n-vector case) that T_k tends towards a diagonal matrix containing the eigenvalues of A.

For the special case of $U_0 = I$ (i.e., n vectors), it is possible to derive a recurrence based on T_k only, leading to the well-known *QR iteration*. Letting $T_0 = A$, we have the very simple two-step iteration for $k = 0, 1, 2, \ldots$:

1. Factor T_k into $T_k = Q_k R_k$, where Q_k is orthogonal and R_k is upper triangular (QR factorization);
2. Compute $T_{k+1} = R_k Q_k$.

We claim that this is equivalent to the orthogonal iteration with $T_k = U_k^T A U_k$. Indeed, since $U_0 = I$, this is satisfied for $k = 0$. We now proceed inductively by assuming the result for k and proving it for $k + 1$. Note that T_k can be written as

$$T_k = U_k^T Z_k = U_k^T U_{k+1} Y_{k+1}.$$

Since $U_k^T U_{k+1}$ is orthogonal and Y_{k+1} is upper triangular, this gives a valid QR factorization of T_k into $Q_k = U_k^T U_{k+1}$ and $R_k = Y_{k+1}$. So the recurrence for T_{k+1} leads to

$$T_{k+1} = R_k Q_k = Y_{k+1} U_k^T U_{k+1} = (U_{k+1}^T Z_k) U_k^T U_{k+1} = U_{k+1}^T A U_{k+1},$$

which completes the induction.

The basic QR method inherits the basic properties of the power method, meaning that the convergence of eigenvalues is linear, with the rate dependent on the gap between eigenvalues, see Example 4.2. Moreover, each iteration requires a QR factorization, which requires $O(n^3)$ arithmetic operations. In order to make the method more efficient and converge faster, practical QR implementations typically use the following tricks:

- A preprocessing step reduces a general (symmetric) matrix A into tridiagonal form, i.e., one finds an orthogonal matrix Q_0 such that $T = Q_0^T A Q_0$ is tridiagonal. This reduces the cost of each step to $O(n)$ instead of $O(n^3)$. Note that T and A have the same eigenvalues, and eigenvectors can be recovered from multiplication by Q_0.
- A shift of the form $A - \mu I$ with μ close to an eigenvalue is often applied explicitly or implicitly in order to speed up convergence. This works because the QR step decouples A into diagonal blocks when A is singular. Once the blocks are decoupled, a divide-and-conquer strategy can be used to solve the smaller subproblems that arise.

Reduction to tridiagonal form. To reduce a symmetric matrix A into tridiagonal form, we use *Householder transformations*, which are orthogonal matrices of the

form

$$H = I - 2\mathbf{v}\mathbf{v}^T$$

with $\|\mathbf{v}\|_2 = 1$. The vector $\mathbf{v}$ is called a *Householder vector*. The first step in the tridiagonal reduction is to find a Householder vector of the form

$$\mathbf{v} = (0, v_2, \ldots, v_n)^T,$$

such that H applied to the first column $\mathbf{a}_1$ of A is non-zero only in its first two entries. In other words, we seek H such that $H\mathbf{a}_1 = a_{11}\mathbf{e}_1 + \alpha\mathbf{e}_2$, where $\mathbf{e}_1$ and $\mathbf{e}_2$ are the first two columns of the identity matrix, respectively. (Note that the first coefficient of $H\mathbf{a}_1$ must be a_{11} because $\langle\mathbf{e}_1\rangle$ and $\langle\mathbf{e}_1\rangle^\perp$ are both invariant subspaces of H.) Using the fact that $H^2 = I$, we calculate

$$\mathbf{a}_1 = H(H\mathbf{a}_1) = (I - 2\mathbf{v}\mathbf{v}^T)(a_{11}\mathbf{e}_1 + \alpha\mathbf{e}_2) = a_{11}\mathbf{e}_1 + \alpha(\mathbf{e}_2 - 2v_2\mathbf{v}).$$

Since H preserves 2-norms, we conclude that $\alpha = \pm\|\mathbf{a}_1 - a_{11}\mathbf{e}_1\|_2$, so that $\mathbf{v}$ can be calculated by

$$\mathbf{w} = \mathbf{e}_2 \pm \frac{\mathbf{a}_1 - a_{11}\mathbf{e}_1}{\|\mathbf{a}_1 - a_{11}\mathbf{e}_1\|_2}, \qquad \mathbf{v} = \frac{\mathbf{w}}{\|\mathbf{w}\|_2},$$

where the plus or minus is chosen so that $w_2 > 1$, which ensures numerical stability. Applying H to the whole matrix A in the sequence $A \to HA \to HAH^T$ leads to the transformation

$$\begin{bmatrix} X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \\ X & X & X & X & X \end{bmatrix} \to \begin{bmatrix} X & X & X & X & X \\ Y & Y & Y & Y & Y \\ 0 & Y & Y & Y & Y \\ 0 & Y & Y & Y & Y \\ 0 & Y & Y & Y & Y \end{bmatrix} \to \begin{bmatrix} X & Y & 0 & 0 & 0 \\ Y & Z & Z & Z & Z \\ 0 & Z & Z & Z & Z \\ 0 & Z & Z & Z & Z \\ 0 & Z & Z & Z & Z \end{bmatrix},$$

where we use a different letter each time an entry is modified by a transformation. Note that the zeros in the first column appear because of the choice of $\mathbf{v}$, and the zeros in the first row appear because of symmetry. We can now apply the same reduction recursively to the lower part (the 'Z' part) of the matrix, and after $n-2$ steps, we obtain a matrix in tridiagonal form.

QR With Shift. To accelerate convergence of the QR method, we apply a shift μ_k that is close to an eigenvalue of A, often chosen to be the bottom rightmost entry of latest iterate T_k. The modified algorithm is to perform, for $k = 0, 1, 2, \ldots$, the following steps:

1. Factor $T_k - \mu_k I = Q_k R_k$ (QR factorization);
2. Compute $T_{k+1} = R_k Q_k + \mu_k I$.

Example 4.3. *Consider the* 2×2 *matrix*

$$T_0 = A = \begin{bmatrix} 2 & \epsilon \\ \epsilon & 1 \end{bmatrix}.$$

One step of the QR with no shift gives

$$Q = \frac{1}{\sqrt{4+\epsilon^2}} \begin{bmatrix} 2 & -\epsilon \\ \epsilon & 2 \end{bmatrix}, \quad T_1 = Q^T A Q = \begin{bmatrix} * & \textit{sym.} \\ \frac{\epsilon(2-\epsilon^2)}{4+\epsilon^2} & * \end{bmatrix}.$$

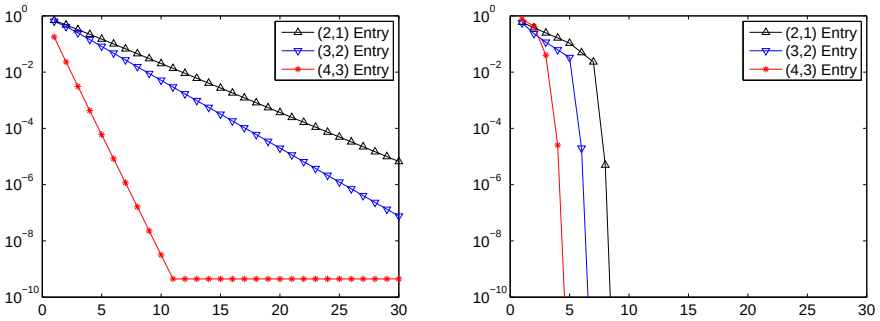

FIGURE 11. Convergence of off-diagonal entries for the 4×4 matrix in Example 4.2. Left: Basic (unshifted) QR iteration. Right: QR method with shift.

We see that the off-diagonal entry is multiplied approximately by 1/2 when ϵ is small, so the method converges approximately linearly with factor 1/2. On the other hand, if we use a shift of $\mu_0 = 1$, we get

$$Q = \frac{1}{\sqrt{1+\epsilon^2}}\begin{bmatrix} 1 & -\epsilon \\ \epsilon & 1 \end{bmatrix}, \quad T_1 = Q^T A Q = \begin{bmatrix} * & \textit{sym.} \\ -\frac{\epsilon^3}{1+\epsilon^2} & * \end{bmatrix}.$$

The off-diagonal entry is now $O(\epsilon^3)$, which indicates cubic convergence. *Thus, if ϵ is small (say ≈ 0.1), the shifted version converges much faster than the basic version. The same observation can be made for larger matrices, see Figure 11 for the convergence of off-diagonal entries for the 4×4 matrix in Example 4.2.*

Remark. The QR iteration (or, more precisely, its practical variants) is the workhorse for small and/or dense eigenvalue problems, with robust and efficient implementations widely available, e.g. in LAPACK. For more details, see [21, 23, 32].

4.3. **Bisection.** In this section, we introduce the *bisecton* method, which gives a different way of computing eigenvalues of a symmetric matrix in tridiagonal form. In addition to approximating specific eigenvalues, this method has the interesting property of revealing the number of eigenvalues that lie within a given interval (λ_-, λ_+); this can be useful in engineering design, for instance, where one would need to ensure a structure has no resonant frequencies within a given range.

We start by considering the $n \times n$ tridiagonal matrix

$$T_n = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{n-1} \\ 0 & & & \beta_{n-1} & \alpha_n \end{bmatrix}.$$

as well as its *principal submatrices* $T_j = T_n(1:j, 1:j)$, $j = 1, \ldots, n-1$. Such a family of tridiagonal matrices enjoy the following properties:

1. The characteristic polynomials $p_j(\lambda)$ of T_j satisfy the recurrence

$$p_j(\lambda) = (\alpha_j - \lambda)p_{j-1}(\lambda) - \beta_{j-1}^2 p_{j-2}(\lambda), \quad p_0(\lambda) = 1. \tag{11}$$

Proof: Direct expansion of $\det(T_n - \lambda I_n)$*, starting with the bottom row.*

2. T_n has no repeated eigenvalues if it is *unreduced*, i.e., if $\beta_i \neq 0$ for all i. *Proof: If repeated eigenvalues exist, then* p_n *and* p_{n-1} *have a common linear factor, which means* p_{n-1} *and* p_{n-2} *have a common linear factor, etc., leading to a contradiction because* $p_0 \equiv 1$ *has no linear factors.*

3. If $\lambda_1^{(j)} \leq \cdots \leq \lambda_j^{(j)}$ are the eigenvalues of T_j for $j = 1, \ldots, n$, then we have the following *interlacing property*:

$$\lambda_1^{(j+1)} \leq \lambda_1^{(j)} \leq \lambda_2^{(j+1)} \leq \cdots \leq \lambda_k^{(j)} \leq \lambda_{k+1}^{(j+1)}.$$

Proof: Use the minimax principle.

Suppose we wish to calculate $\lambda_k^{(n)}$, the k-th eigenvalue of the original matrix T_n. The following theorem will be useful for determining where this occurs in the spectrum.

Theorem 4.1 (Sturm Sequence Property)**.** *Let* T_n *be a symmetric, unreduced tridiagonal matrix, and let* $\omega(\lambda)$ *be the number of sign changes in the Sturm sequence* $(p_0(\lambda), p_1(\lambda), \ldots, p_n(\lambda))$*. Then* $\omega(\lambda)$ *equals the number of eigenvalues of* T_n *less than* λ*.*

The proof uses the recurrence (11) and a counting argument, and can be found in [45, pp. 300–301]. Since the Sturm sequence is easy to evaluate for a given λ using the recurrence (11), an approximation of $\lambda_k^{(n)}$ can be computed as follows:

Bisection method for finding $\lambda_k^{(n)}$*:*

1. Find y and z such that $y < \lambda_k^{(n)} < z$ using e.g. Gershgorin's Theorem, cf. Section 7.3, Exercise 3.
2. While $|z - y| > tol$, do
   - Compute $x = (y + z)/2$.
   - If $\omega(x) \geq k$, set $z := x$; else set $y := x$.

Remarks.

1. In order to avoid overflow when n is large, it is sometimes preferable to adapt (11) to compute $\varphi_i(\lambda) := p_i(\lambda)/p_{i-1}(\lambda)$ rather than the $p_i(\lambda)$ themselves. In that case, one would count the number of negative values of $\varphi_i(\lambda)$ instead of the number of sign changes.
2. The bisection method converges approximately linearly with a factor $1/2$, since it halves the interval that containing $\lambda_k^{(n)}$ at every step. Once the interval is small enough to contain a single eigenvalue, it is possible to switch to another method (e.g., shift-and-invert) to obtain faster convergence to $\lambda_k^{(n)}$.

4.4. **Lanczos Method.** For large sparse matrices, the Householder tridiagonalization procedure described in Section 4.2 requires too many operations and becomes impractical. A different way of generating a tridiagonal matrix from which eigenvalues can be calculated is based on the idea of Krylov subspaces. Given a matrix

A and an initial vector $\mathbf{q}_1$, the *Krylov subspace* $\mathcal{K}_r(A;\mathbf{q}_1)$ is defined as

$$\mathcal{K}_r(A;\mathbf{q}_1) = \text{Span}\{\mathbf{q}_1, A\mathbf{q}_1, \ldots, A^r\mathbf{q}_1\}.$$

From this definition, we see that every element of $\mathcal{K}_r(A;\mathbf{q}_1)$ can be written as $P_r(A)\mathbf{q}_1$ for some polynomial $P_r(x)$ of degree $\leq r$. Moreover, we have $\mathcal{K}_r(A;\mathbf{q}_1) \subset \mathcal{K}_{r+1}(A;\mathbf{q}_1)$. If they are equal, then $\mathcal{K}_r(A;\mathbf{q}_1)$ is an *invariant subspace* of A.

As we have seen in the power method, repeated multiplication makes $A^k\mathbf{q}_1$ closer and closer to the largest eigenvector of A, and making the set of factors more and more linearly dependent. Thus, one often creates an *orthonormal* basis for $\mathcal{K}_r(A;\mathbf{q}_1)$ obtained from the so-called *Arnoldi process*, which uses the Gram-Schmidt orthogonalization procedure:[6] for $k = 1, 2, \ldots,$ compute

$$\mathbf{r}_k = A\mathbf{q}_k - \sum_{j=1}^{k} (\mathbf{q}_j^T A\mathbf{q}_k)\mathbf{q}_j, \qquad \mathbf{q}_{k+1} = \frac{\mathbf{r}_k}{\|\mathbf{r}_k\|_2} \quad \text{if } \mathbf{v}_k \neq 0. \tag{12}$$

It is clear that $\{\mathbf{q}_1, \ldots, \mathbf{q}_{r+1}\}$ also spans $\mathcal{K}_r(A;\mathbf{q}_1)$, as long as none of the $\mathbf{r}_k$ vanishes. If $\mathbf{r}_k = 0$ for some k, then $\mathcal{K}_{k-1}(A;\mathbf{q}_1) = \mathcal{K}_k(A;\mathbf{q}_1)$, so we have found an invariant subspace of A. For sparse matrices A, the subspace $\mathcal{K}_r(A;\mathbf{q}_1)$ is easy to generate, since it suffices to multiply A by different vectors repeatedly. In fact, it is not even necessary to store the matrix A itself, as long as a routine exists for performing the required matrix-vector multiplication; such procedures are known as *matrix-free methods*.

The key observation in deriving the Lanczos method is that if A is symmetric, then $A\mathbf{q}_k$ is automatically orthogonal to $\mathbf{q}_1, \ldots, \mathbf{q}_{k-2}$ (prove it!). Thus, the sum in (12) runs only from $j = k-1$ to k. Letting $\beta_k = \|\mathbf{r}_k\|_2$, we can rewrite (12) as the three-term recurrence

$$A\mathbf{q}_k = \beta_{k-1}\mathbf{q}_{k-1} + \alpha_k\mathbf{q}_k + \beta_k\mathbf{q}_{k+1}, \tag{13}$$

where $\alpha_k = \mathbf{q}_k^T A\mathbf{q}_k$. Thus, we have derived the *Lanczos process*: given a starting vector $\mathbf{q}_1$ (and assuming $\mathbf{q}_0 = 0$ and $\beta_0 = 0$), do for $k = 1, 2, \ldots$:

1. Compute $\alpha_k = \mathbf{q}_k^T A\mathbf{q}_k$, $\mathbf{r}_k = (A - \alpha_k I)\mathbf{q}_k - \beta_{k-1}\mathbf{q}_{k-1}$;
2. If $\mathbf{r}_k \neq 0$, normalize to get $\mathbf{q}_{k+1} = \mathbf{r}_k/\|\mathbf{r}_k\|$, $\beta_k = \|\mathbf{r}_k\|$.

The Lanczos process is said to *break down* if $\mathbf{r}_k = 0$, or equivalently, if $\beta_k = 0$.

Another way of writing (13) after k steps of Lanczos is the matrix relation

$$AQ_k = Q_k T_k + \beta_{k+1}\mathbf{q}_{k+1}\mathbf{e}_k^T, \tag{14}$$

where $Q_k = [\mathbf{q}_1, \ldots, \mathbf{q}_k]$ has orthonormal columns, $\mathbf{e}_k$ is the k-th column of the identity matrix, and

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \ddots & & 0 \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{k-1} \\ 0 & & & \beta_{k-1} & \alpha_k \end{bmatrix}.$$

[6] In fact, the formula (12) suffers from loss of orthogonality when A is ill-conditioned. In that case, a modified Gram-Schmidt or Householder orthogonalization is preferred, see [38].

Observe that by multiplying (14) on the left by Q_k^T and using the fact that $\mathbf{q}_{k+1}$ is orthogonal to all previous $\mathbf{q}_j$, we get

$$Q_k^T A Q_k = T_k. \tag{15}$$

A crucial question is how well the eigenvalues of T_k, also known as the "Ritz values" of A, approximate the eigenvalues of A. Indeed, in the case of breakdown at step k, we have the exact relation

$$AQ_k = Q_k T_k,$$

so Q_k generates an invariant subspace of A, and the spectrum of T_k is a subset of that of A. If $\mathbf{r}_k \neq 0$, then there are many results describing the relationship between the eigenvalues of T_k and A, collectively known as *Kaniel–Page Theory*. Below we attempt to give a flavor of this type of analysis.

Let us estimate the convergence of *extremal* Ritz values, i.e., the smallest and largest eigenvalues of T_k, towards the extremal eigenvalues of A as a function of k, the number of Lanczos steps. Let $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of A, and $\tilde{\lambda}_1 \leq \tilde{\lambda}_2 \leq \cdots \leq \tilde{\lambda}_k$ be those of $T_k = Q_k^T A Q_k$. Then from the minimax principle, we know that

$$\tilde{\lambda}_1 = \min_{\substack{\mathbf{x}\in\mathbb{R}^k \\ \mathbf{x}\neq 0}} \frac{\mathbf{x}^T Q_k^T A Q_k \mathbf{x}}{\mathbf{x}^T \mathbf{x}} = \min_{\substack{\mathbf{w}\in\mathcal{K}_{k-1}(A,\mathbf{q}_1) \\ \mathbf{w}\neq 0}} \frac{\mathbf{w}^T A \mathbf{w}}{\mathbf{w}^T \mathbf{w}} \geq \lambda_1.$$

In order to estimate this minimum, observe that for any $\mathbf{w} \in \mathcal{K}_{k-1}(A, \mathbf{q}_1)$, there exists a polynomial $P_{k-1}(x)$ of degree $k-1$ or lower such that $\mathbf{w} = P_{k-1}(A)\mathbf{q}_1$. Because $\tilde{\lambda}_1$ minimizes the Rayleigh quotient over all such polynomials, we in fact have

$$\tilde{\lambda}_1 \leq \frac{\sum_{j=1}^n \alpha_j^2 \lambda_j P_{k-1}^2(\lambda_j)}{\sum_{j=1}^n \alpha_j^2 P_{k-1}^2(\lambda_j)} = \lambda_1 + \frac{\sum_{j=2}^n (\lambda_j - \lambda_1)\alpha_j^2 P_{k-1}^2(\lambda_j)}{\sum_{j=1}^n \alpha_j^2 P_{k-1}^2(\lambda_j)},$$

where $\mathbf{q}_1 = \sum_{j=1}^n \alpha_j \mathbf{v}_j$ is the eigen-decomposition of $\mathbf{q}_1$. This is true for any polynomial $P_{k-1}(x)$, but we would like to choose a specific polynomial so that the second term on the right-hand side is small. To do so, we choose P_{k-1} to be a *shifted Chebyshev polynomial* that is small on the interval $[\lambda_2, \lambda_n]$ but large at λ_1. Recall that Chebyshev polynomials are defined by the recurrence

$$f_0(x) = 1, \qquad f_1(x) = x, \qquad f_{n+1}(x) = 2x f_n(x) - f_{n-1}(x), \quad n \geq 1$$

and satisfy

$$f_n(x) = \begin{cases} \cos(n \arccos(x)), & |x| \leq 1, \\ (\operatorname{sgn}(x))^n \cosh(n \operatorname{arcosh}|x|), & |x| > 1. \end{cases}$$

In other words, we have $|f_n(x)| \leq 1$ for $x \in [-1, 1]$, and $|f_n(x)|$ grows like $2^{n-1}|x|^n$ outside this interval. Thus, the shifted Chebyshev polynomial

$$P_{k-1}(\lambda) = f_{k-1}\left(\frac{2\lambda - (\lambda_2 + \lambda_n)}{2(\lambda_n - \lambda_2)}\right)$$

is small on $[\lambda_2, \lambda_n]$, but large at λ_1, provided that it is far enough from λ_2. Using the same reasoning for the largest eigenvalue λ_n, we obtain the following convergence result.

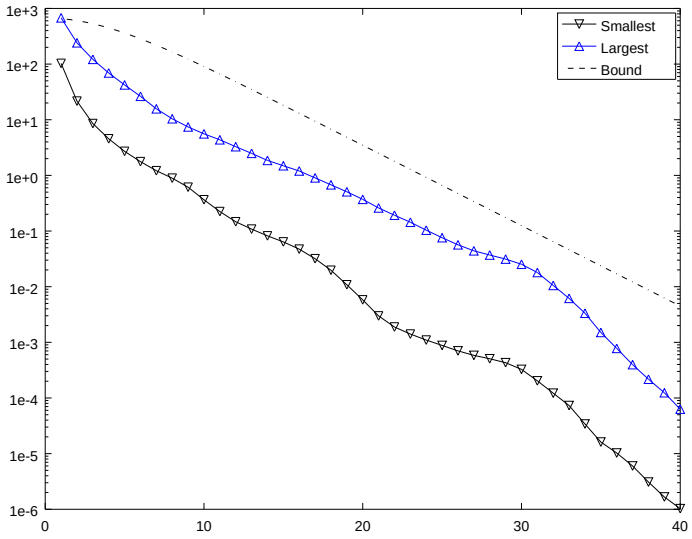

FIGURE 12. Lanczos method applied to the finite-difference Laplacian for an L-shaped domain, $h = 0.1$

Theorem 4.2 (Convergence of Lanczos)**.** *In exact arithmetic, the extremal Ritz values $\tilde{\lambda}_1$ and $\tilde{\lambda}_k$ after k iterations of Lanczos satisfy the estimate*

$$\lambda_1 \le \tilde{\lambda}_1 \le \lambda_1 + C_1/\cosh^2((k-1)\gamma_1), \tag{16}$$

$$\lambda_n \ge \tilde{\lambda}_k \ge \lambda_n - C_2/\cosh^2((k-1)\gamma_2), \tag{17}$$

where

$$\gamma_1 = \operatorname{arcosh}\left(\frac{\lambda_2 + \lambda_n - 2\lambda_1}{\lambda_n - \lambda_2}\right),\ \gamma_2 = \operatorname{arcosh}\left(\frac{2\lambda_n - \lambda_1 - \lambda_{n-1}}{\lambda_{n-1} - \lambda_1}\right).$$

In Figure 12, we plot the error in the largest and smallest Ritz values and the theoretical bound given in Theorem 4.2 for the finite-difference discrete Laplacian for an L-shaped domain. We see that this simple bound is not the sharpest, but nonetheless tracks the general slope of the error curves reasonably well.

Similar techniques can be used to estimate interior eigenvalues, although such bounds may contain constants that become quite large. For instance, we have the estimate

$$\lambda_i \le \tilde{\lambda}_i \le \lambda_i + C\kappa_i/\cosh^2((k-i)\gamma_i),$$

where $\gamma_i = \operatorname{arcosh}\left(\frac{\lambda_{i+1}+\lambda_n-2\lambda_i}{\lambda_n-\lambda_{i+1}}\right)$, $\kappa_i = \prod_{j=1}^{i-1}\left(\frac{\lambda_n-\tilde{\lambda}_j}{\lambda_i-\tilde{\lambda}_j}\right)^2$.

Remarks.

1. Just as for power-type methods, the *extremal* eigenvalues (largest and smallest) are among the first to converge in the Lanczos method.
2. Because the short recurrence (13) does not enforce orthogonality explicitly with all previous $\mathbf{q}_i$, orthogonality may be lost gradually due to round-off errors in finite precision arithmetic. When this happens, the subsequent

$\mathbf{q}_i$ may be (nearly) linearly dependent on the previous ones, giving rise to repeated *ghost eigenvalues*. To prevent this, one can enforce orthogonality explicitly, either by complete re-orthogonalization (expensive), or by selective re-orthogonalization against "converged" Ritz vectors only. For more details, see [23].

3. Lanczos is intimately related to the celebrated Conjugate Gradient method for solving $A\mathbf{x} = \mathbf{b}$. See [23, 38] for more details.

5. Application: Vibrating Plates

Chladni figures are patterns that were discovered by Ernst Florence Friedrich Chladni (1756–1827) in a series of experiments on vibrating plates. In one of these experiments, he sprinkles fine sand onto a square plate before setting it into vibration using a violin bow. As the plate vibrates, the sand gathers at points that remain stationary and traces out beautiful patterns, such as the ones shown in Figure 13. It turns out such patterns are zero sets of eigenfunctions of the biharmonic operator, with special boundary conditions that can be derived from a variational formulation. In this section, we will show how this eigenvalue problem can be derived, discretized and calculated using techniques we have seen. The presentation of this section follows [20] closely.

5.1. **Vibrating plate model.** In order to derive the vibrating plate model, we start just like we did in Section 1, with Newton's second law:

$$\frac{\partial^2 z}{\partial t^2} = -\mathscr{L} z.$$

We saw that such motions can be analyzed using standing wave solutions of the form $z(x, y, t) = T(t)u(x, y)$ is a standing wave, where $u(x, y)$ is a solution of the eigenvalue problem

$$\mathscr{L} u = \lambda u.$$

In the absence of external forces, $\mathscr{L} u$ is simply the restoring force due to the shape of the vibrating plate $u = u(x, y)$. However, instead of $-\mathscr{L}$ being the Laplacian operator, as was the case for vibrating membranes, we will deduce the form of $\mathscr{L}$ based on energy considerations. Gustav Kirchhoff (1824–1887) proposed a model that describes the *potential energy* stored in a deformed thin square plate with shape $u : \Omega \to \mathbb{R}$, given by

$$J[u(x, y)] = \frac{1}{2} \iint_\Omega (u_{xx} + u_{yy})^2 - 2(1 - \mu)(u_{xx}u_{yy} - u_{xy}^2)\, dx\, dy, \tag{18}$$

where $0 < \mu < 1$ is a material constant. This is only one of many models for the bending plate, and it may not be adequate for certain types of problems; see the plate paradox in [7], further explored in [6]. Other possibilities include the Reissner-Mindlin model or the full 3D model, as described in [6]. However, for simplicity, we will only consider the Kirchhoff model (18) in these notes.

To see how this energy functional (18) is related to the operator $\mathscr{L}$, we resort to arguments that are standard in the calculus of variations, see [22, Chapter 7]: suppose we wish to change the shape of the plate from an initial configuration $u(x, y)$ to a slighly different shape $u(x, y) + \varepsilon v(x, y)$, where ε is small. Then the change in potential energy is precisely the work done on the system, which is the distance travelled by each particle on the plate times the force needed to counteract the restoring force $\mathscr{L} u(x, y)$, see Figure 14. In other words, we have

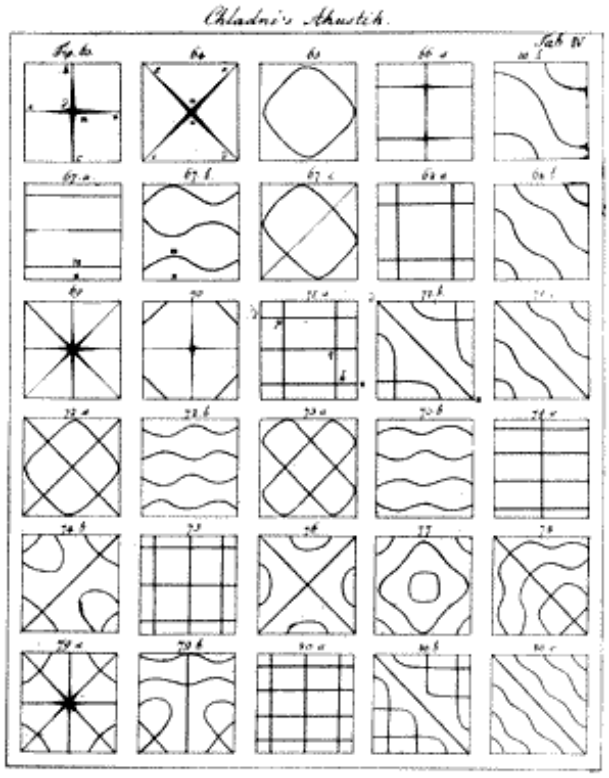

FIGURE 13. Chladni figures as recorded by Chladni in [16].

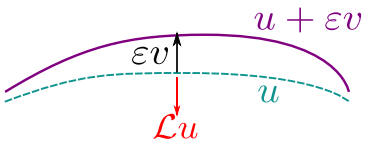

FIGURE 14. Forces involved in deforming of a thin plate from configuration u to a nearby configuration $u + \varepsilon v$.

$$J[u + \varepsilon v] - J[u] = \iint_\Omega (\mathscr{L}u)(\varepsilon v)\,dx\,dy + O(\varepsilon^2).$$

Dividing both sides by ε and taking the limit as $\varepsilon \to 0$, we obtain

$$\frac{d}{d\varepsilon} J[u + \varepsilon v]\Big|_{\varepsilon=0} = \lim_{\varepsilon \to 0} \frac{J[u + \varepsilon v] - J[u]}{\varepsilon} = \iint_\Omega (\mathscr{L}u)v\,dx\,dy.$$

Substituting the above into the definition (18) and differentiating yields

$$\iint_\Omega (\mathscr{L}u)v\,dx\,dy = \iint_\Omega (u_{xx}+u_{yy})(v_{xx}+v_{yy}) - (1-\mu)(u_{xx}v_{yy}+u_{yy}v_{xx}-2u_{xy}v_{xy})\,dx\,dy.$$

This is precisely the expression we need to define the *weak form* of the eigenvalue problem

$$\iint_\Omega (\mathscr{L}u)v\,dx\,dy = \iint_\Omega \lambda uv\,dx\,dy.$$

We will use this form in Section 5.3 to derive a *spectral approximation* of the eigenvalue problem, as was done by Walther Ritz [36].

For the moment, we continue by deriving the strong form of the problem for the special case of a rectangular plate, where $\Omega = (-L, L) \times (-H, H)$. We integrate by parts to eliminate derivatives of v:

$$\begin{aligned}\iint_\Omega (\mathscr{L}u)v\,dx\,dy = \iint_\Omega (u_{xxxx}+2u_{xxyy}+u_{yyyy})v\,dx\,dy \\ + \int_{x=\pm L}\Big([u_{xx}+\mu u_{yy}]\frac{\partial v}{\partial n} + [u_{xxx}+(2-\mu)u_{xyy}]v\Big)\,dy \\ + \int_{y=\pm H}\Big([\mu u_{xx}+u_{yy}]\frac{\partial v}{\partial n} + [u_{yyy}+(2-\mu)u_{xxy}]v\Big)\,dx \\ + 2(1-\mu)[u_{xy}(L,H)v(L,H) - u_{xy}(-L,H)v(-L,H) \\ - u_{xy}(L,-H)v(L,-H) + u_{xy}(-L,-H)v(-L,-H)]\end{aligned}$$

By choosing arbitrary functions v that vanish on the boundary of Ω, we obtain the PDE

$$u_{xxxx} + 2u_{xxyy} + u_{yyyy} = \lambda u, \tag{19}$$

which must hold at every point in the interior of Ω. We can also obtain boundary conditions by choosing arbitrary variations along different parts of the boundary. For instance, by considering the integrals along $x = \pm L$, we obtain the boundary conditions

$$u_{xx} + \mu u_{yy} = 0, \qquad u_{xxx} + (2-\mu)u_{xyy} = 0$$

there. We can combine the above with the similar-looking conditions along the horizontal boundaries $y = \pm H$ by writing

$$u_{nn} + \mu u_{\tau\tau} = 0, \qquad u_{nnn} + (2-\mu)u_{n\tau\tau} = 0, \tag{20}$$

where the subscripts n and τ denote derivatives in the outward normal and tangential directions, respectively. Finally, by taking arbitrary variations at the four corners, we obtain the corner conditions

$$u_{xy}(\pm L, \pm H) = 0. \tag{21}$$

Equations (20) and (21) are known as *free boundary conditions* for the thin plate PDE (19). Another way of writing (19) is $\Delta^2 u = \lambda u$, where Δ is the usual Laplacian operator; Δ^2 is also known as the *biharmonic operator*. Note that the material constant μ does not enter into the PDE itself, but instead appears in the boundary conditions. The PDE (19) has already appeared in a series of articles by Sophie Germain in 1811–21; however, the edge conditions (20) only appears in 1850 in the

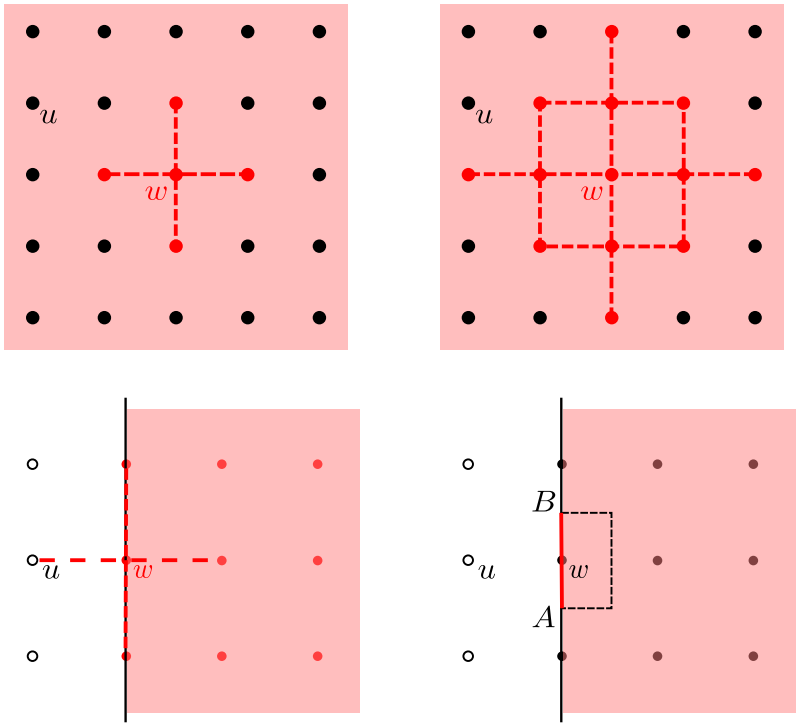

FIGURE 15. Finite difference/finite volume discretization of the square plate problem. Top left: definition of w. Top right: full stencil of the biharmonic operator after composing the five-point stencil with itself. Bottom left: elimination of ghost points using second order boundary conditions. Bottom right: treatment of third order boundary conditions.

work by Kirchhoff, and the corner conditions (21) finally emerge in an article by Lamb in 1889, almost 80 years after the discovery of the strong form PDE.

5.2. **Finite Difference Discretization.** In order to discretize the strong form (19)–(21), we use the finite difference and finite volume techniques introduced in Section 2. We show this for the case $L = H = 1$, i.e., for the square $\Omega = (-1, 1)^2$. Let $(u_{ij})_{i,j=0}^N$ be the grid points distributed uniformly at $(x_i, y_j) = (-1 + ih, -1 + jh)$, $h = 2/N$. Since the biharmonic operator can be regarded as the Laplacian operator composed with itself, it is natural to discretize it by composing the discrete five-point Laplacian stencil with itself. If we define the auxiliary grid function w using the five-point stencil as shown in the top left panel of Figure 15, i.e.,

$$w_{ij} := \frac{1}{h^2}(4u_{ij} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}) \approx -\Delta u,$$

then (19) can be discretized i as

$$\frac{1}{h^2}(4w_{ij} - w_{i-1,j} - w_{i+1,j} - w_{i,j-1} - w_{i,j+1}) = \lambda u_{ij}.$$

This leads to the wide stencil shown in the top right panel of Figure 15, which can be written for the interior points $1 \le i, j \le N-1$.

Note that if $i = 1$ or $N-1$, the stencil involves the unknown $u_{-1,j}$ or $u_{N+1,j}$, which falls outside Ω; this is a *ghost point* that needs to be eliminated. To do so, we make use of the second order boundary condition in (20): along $x = -1$, the condition reads $u_{xx} + \mu u_{yy} = 0$, which can be discretized as

$$2(1+\mu)u_{0,j} - u_{-1,j} - u_{1,j} - \mu(u_{0,j-1} + u_{0,j+1}) = 0.$$

This allows us to represent the ghost value as

$$u_{-1,j} = 2(1+\mu)u_{0,j} - u_{-1,j} - u_{1,j} - \mu(u_{0,j-1} + u_{0,j+1}),$$

as shown in the bottom left panel of Figure 15. Other ghost points can be treated similarly.

Finally, we need equations for the edge variables themselves, which we will derive using the finite volume method. Consider the edge unknown shown in the bottom right panel of Figure 15. We integrate (19) over the half volume V indicated in dotted lines. Noting that $\Delta^2 u = -\Delta w$, we have

$$-\int_V \Delta w \, dx \, dy = \lambda \int_V u \, dx \, dy \approx \frac{h^2\lambda}{2} u_{ij}.$$

Applying the divergence theorem on the left hand side yields

$$-\int_{\partial V} \frac{\partial w}{\partial n} \, ds \approx \frac{h^2\lambda}{2} u_{ij}.$$

The integral along three of the four edges can be approximated using finite differences. For instance, the integral along the top edge can be approximated by $\frac{1}{2}(w_{0,j} - w_{0,j+1})$. To deal with the edge along the domain boundary, we use the third order boundary condition to eliminate $\frac{\partial w}{\partial n}$: since $w_x = -(u_{xxx} + u_{xyy})$, we have by (20)

$$w_x = -(\underbrace{u_{xxx} + (2-\mu)u_{xyy}}_{=0}) + (1-\mu)u_{xyy}.$$

Thus, the integral along the boundary portion of ∂V is given by

$$-\int_A^B \frac{\partial w}{\partial n} \, dy = \int_A^B w_x \, dy = (1-\mu)\int_A^B u_{xyy} \, dy = (1-\mu)(u_{xy}(B) - u_{xy}(A)).$$

The mixed derivative at B can now be approximated by a finite difference involving $u_{-1,j+1}$, $u_{-1,j}$, $u_{1,j+1}$ and $u_{1,j}$, and similarly for the derivative at A. Finally, for corner points, we use the same technique over the quarter volume, with one of the mixed derivatives vanishing directly because of the corner condition (21).

The above discretization leads to a large, sparse *generalized* eigenvalue problem of the form

$$Au = \lambda B u,$$

where B is a diagonal matrix containing the areas of the control volumes, and A having at most 13 non-zero entries per row. This matrix is easy to set up in MATLAB, requiring only about 80 lines of code, see [20] for details. The generalized eigenvalue problem can then be solved using a modified version of the Lanczos method, see Section 7.3. If we plot the nodal lines of these eigenfunctions, we obtain the Chladni figures shown in Figure 16.

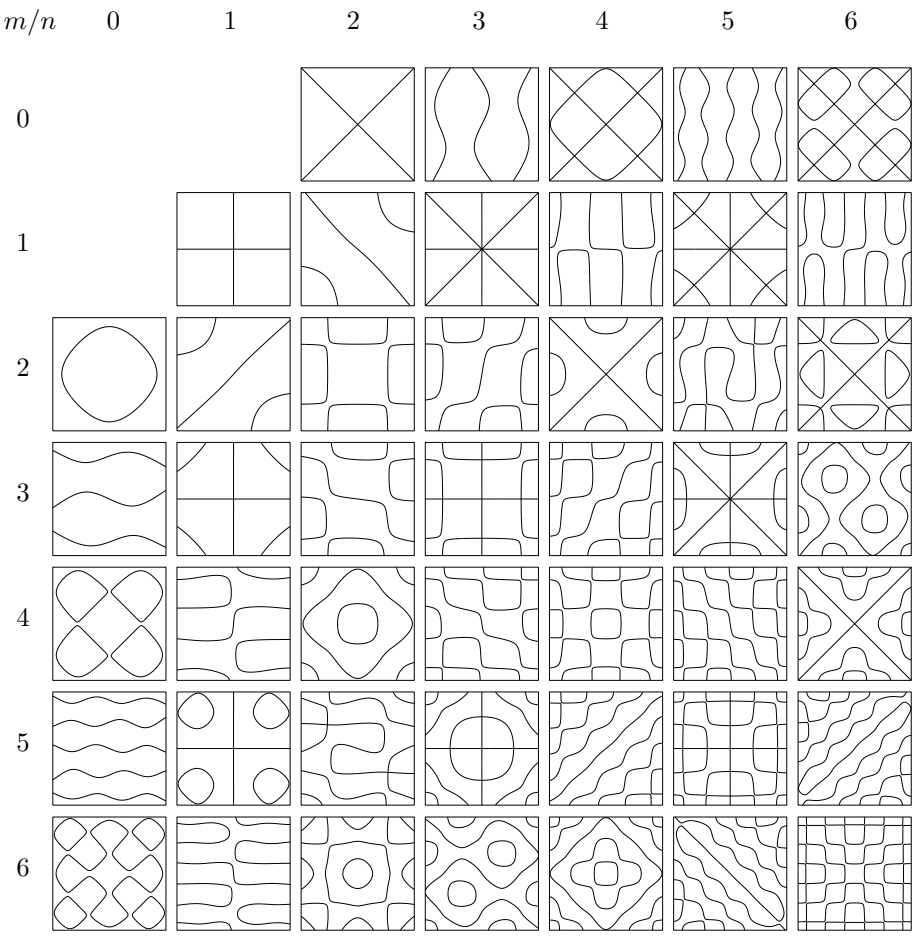

FIGURE 16. Chladni Figures obtained by Matlab using the finite difference method.

5.3. **Ritz Approximation.** Another way of approximating the eigenvalues of the free vibrating plate, developed by Walther Ritz [36], bears much similarity to the finite element method introduced in Section 3.[7] Consider once again the eigenvalue problem in weak form, where we seek $u \in H^2(\Omega)$ such that

$$a(u, v) = \lambda(u, v) \quad \text{for all } v \in H^2(\Omega), \tag{22}$$

where

$$a(u, v) = \iint_\Omega (u_{xx} + u_{yy})(v_{xx} + v_{yy}) - (1 - \mu)(u_{xx}v_{yy} + u_{yy}v_{xx} - 2u_{xy}v_{xy})\, dx\, dy,$$

and $(\cdot, \cdot)$ denotes the L^2 inner product on Ω. Instead of using a basis of hat functions, Ritz's idea is to use a product of one-dimensional eigenfunctions. Suppose

[7] In fact, the finite element method has its origin in Ritz's method.

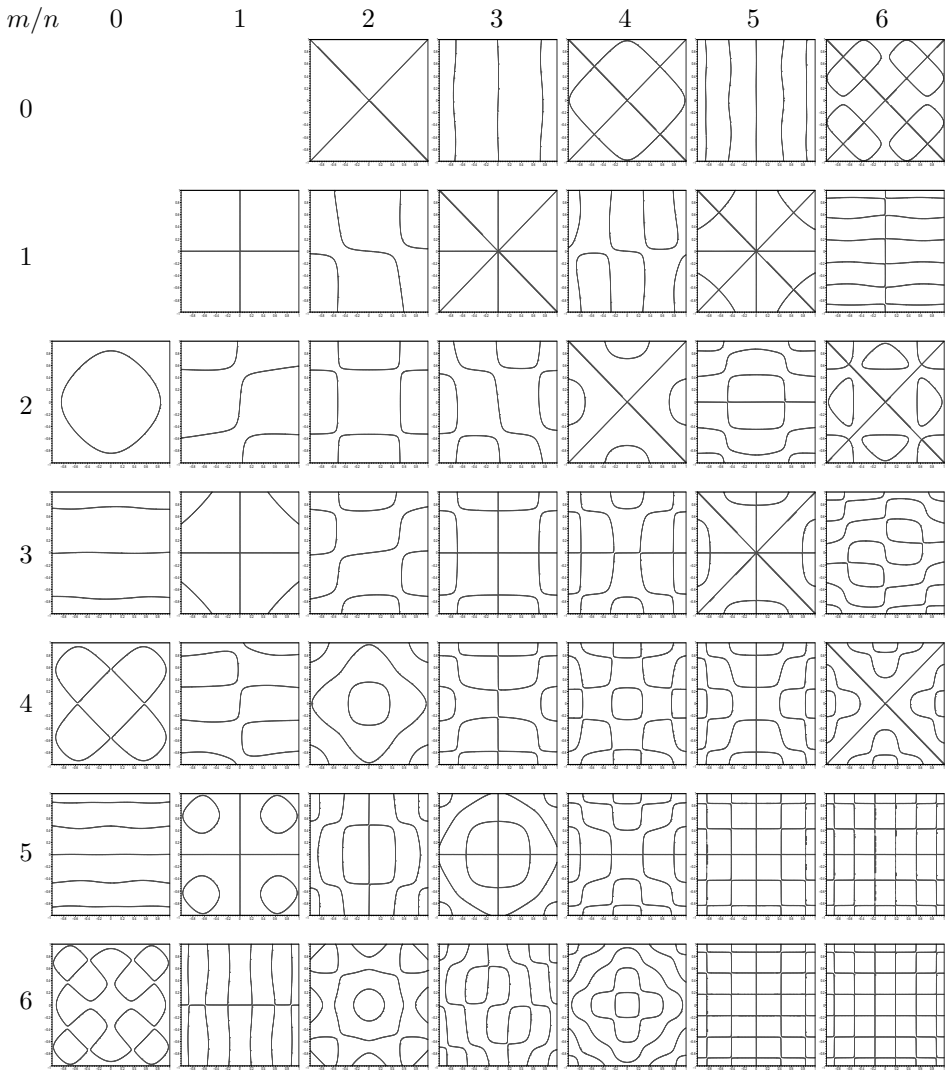

FIGURE 17. Chladni figures obtained by Ritz's method for $N = 6$, i.e., using 49 basis functions.

$u_m(x)$ satisfies the one-dimensional eigenvalue problem

$$\frac{d^4u_m}{dx^4} = k_m^4 u_m, \qquad \frac{d^2u_m}{dx^2} = \frac{d^3u_m}{dx^3} = 0 \text{ at } x = \pm 1.$$

This two point boundary value problem can be solved easily, and we find

(23)

$$u_m(x) = \begin{cases} \dfrac{\cosh k_m \cos k_m x + \cos k_m \cosh k_m x}{\sqrt{\cosh^2 k_m + \cos^2 k_m}}, & \tan k_m + \tanh k_m = 0, \quad m \text{ even}, \\ \dfrac{\sinh k_m \sin k_m x + \sin k_m \sinh k_m x}{\sqrt{\sinh^2 k_m - \sin^2 k_m}}, & \tan k_m - \tanh k_m = 0, \quad m \text{ odd}. \end{cases}$$

We now let the finite dimensional subspace V_h to be spanned by functions of the form $u_m(x)u_n(y)$; in other words, for a given $N > 0$, we seek an approximate eigenfunction of the form

$$u(x,y) = \sum_{m=0}^{N}\sum_{n=0}^{N} c_{mn}u_m(x)u_n(y) \tag{24}$$

such that (22) is satisfied for all $v(x,y) = u_p(x)u_q(y)$, $0 \le p,q \le N$. In matrix form, this gives the generalized eigenvalue problem

$$A\mathbf{c}_k = \tilde{\lambda}_k B\mathbf{c}_k,$$

where A and B are symmetric $(N+1)^2 \times (N+1)^2$ matrices, with rows indexed by (p,q) and columns indexed by (m,n), corresponding to the basis functions $u_{pq}(x,y) := u_p(x)u_q(y)$ and $u_{mn}(x,y) := u_m(x)u_n(y)$ respectively. The corresponding entries are

$$a_{mn}^{pq} = a(u_{mn}, u_{pq}), \qquad b_{mn}^{pq} = (u_{mn}, u_{pq}).$$

Note that the matrices are dense; the calculation of the entries, while cumbersome, can be automated by software such as Maple; see [20] for details. Once A and B are evaluated numerically, the QR method can be used to obtain the eigenvalues $\tilde{\lambda}_k$, and the eigenvectors $\mathbf{c}_k$ can be substituted back into (24) to yield the approximate eigenfunctions. The nodal lines of the latter are shown in Figure 17 for the case with $N = 6$, i.e., when the matrices are 49×49. We see that even for this small number of degrees of freedom, the approximation is already very good for the low frequency modes. Thus, thanks to a clever choice of basis functions, this method converges much faster to the exact eigenvalues in terms of number of degrees of freedom than the finite difference method, at the price of dealing with dense rather than sparse matrices.

Comparing the Chladni figures in 17 with those obtained by Matlab in 16, we observe that there are substantial qualitative differences between them in two cases:

- When $m + n$ is odd. This is because these correspond to eigenvalues with multiplicity 2: the (m,n) and (n,m) modes together span a two-dimensional subspace with two linearly independent eigenfunctions, so the Chladni figures are not uniquely determined. If we take linear combinations of the eigenfunctions obtained by Matlab, we obtain the ones found by Maple. We show an example in the left and middle panels of Figure 18 for the case of $(m,n) = (1,2)$ and $(2,1)$: denoting by $u^{(12)}$ and $u^{(21)}$ the eigenfunctions shown in positions (1,2) and (2,1) in Figure 16, the left panel shows the the linear combination $u = -1.6u^{(12)} + u^{(21)}$, whereas the middle panel shows $v = u^{(12)} + 1.6u^{(21)}$. These look qualitatively similar to the ones found by Maple in positions (1,2) and (2,1) in Figure 17.
- When m and n are large relative to the number of basis functions used, for example when $(m,n) = (5,5)$. When we increase the number of basis functions, e.g., with $N = 9$, we get a much better approximation that matches the Chladni figure found by Matlab, see the right panel of Figure 18.

From the Minimax principle, we know the Ritz values $\tilde{\lambda}_k$ are overestimates of the exact eigenvalues. In Table 2, we compare the first 10 eigenvalues obtained by the finite difference method with 400 grid points per side with those from the Ritz

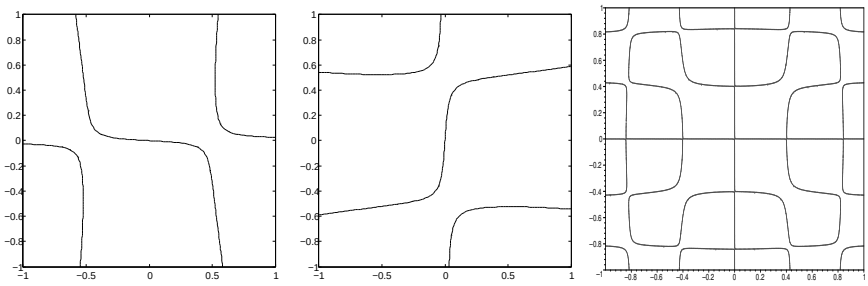

FIGURE 18. Left and middle: nodal lines for linear combinations of the eigenfunctions calculated by Matlab. Right: A better approximation of the eigenfunction obtained from Maple for $m = 5$ and $n = 5$ with 10 basis functions per direction ($0 \le m, n \le 9$).

TABLE 2. First 10 eigenvalues obtained by the finite difference and Ritz methods, versus experimental values found by Chladni. Up/down arrows mean that the observed notes are higher/lower than the indicated nominal values.

Model			Experiment	
Fin. Diff.	Ritz	Gap	Note	Value
12.5	12.5	0.08%	G	12.4
26.0	26.1	0.46%	D↓	26.4
35.6	35.8	0.36%	E↑	36.2
80.9	81.2	0.36%	B	77.5
235.4	236.3	0.39%	G♯ ↓	215
269.3	270.0	0.24%	A↑	260
320.7	322.4	0.53%	B	310
375.2	377.2	0.53%	C↑	364
730.0	732.4	0.33%	F♯	698
876.1	879.8	0.42%	G↑	819

method using $N = 10$ modes per direction. We see that the Ritz values are indeed larger than the finite difference eigenvalues, but they are already very accurate, with a gap no larger than about half a percent between the two. In the same table, we also show the frequencies observed experimentally by Chladni himself in [16], as well as the corresponding eigenvalues. (Recall from Section 1 that the square root of the eigenvalue is the frequency of oscillation, which produces a sound that Chladni heard and recorded as a musical note.) We see that the experimental results also match the model very well, but they tend to be underestimates. This is possibly due to the internal friction present in physical systems, which causes energy to dissipate and leads to lower frequencies when compared with ideal models.

5.4. **Other boundary conditions.** The vibrating plate model given by the energy functional (18) can be used to obtain other boundary conditions. For example, one is said to apply *clamped boundary conditions* when the edge is not allowed to move, and the plate may not pivot around the edge. In the language of the calculus of variation, this is the same as specifying that perturbations of the form $u + \varepsilon v$ are

only allowed if

$$v = \frac{\partial v}{\partial n} = 0.$$

This leads to strong form boundary conditions

$$u = \frac{\partial u}{\partial n} = 0$$

in the corresponding eigenvalue problem. Note that this is the analogue of essential boundary conditions for the Laplacian operator, in the sense that such boundary conditions must be built into the test and trial spaces in the variational problem.

Another common type of boundary conditions is the *simply supported* boundary conditions, where the edge of the plate is not allowed to move, but the plate may pivot around the edge. Thus, any perturbations of the form $u + \varepsilon v$ must satisfy

$$v = 0.$$

This leads to strong form boundary conditions

$$u = 0, \quad \frac{\partial^2 u}{\partial n^2} = 0.$$

Whereas the first condition is rather obvious, the second-order condition is less so – it can be derived by integration by parts and a variational argument, in the same way we deduced the edge and corner conditions for the free boundary conditions. The reader is invited to calculate the eigenvalues and nodal lines of the biharmonic operator for these types of boundary conditions.

6. Further Reading

There is a large body of literature on the numerical solution of PDE eigenvalue problems. Instead of attempting to give a (necessarily non-exhaustive) list of references, we will just give pointers to a few classical and recent works on the subject, through which the interested reader can continue to explore via the references therein.

The approximation of eigenvalue problems was first done by finite difference methods, with much important work appearing before the 1970s, such as [42, 43, 25, 26]. Such methods can be used to derive lower as well as upper bounds on the exact eigenvalues, but because a minimax type argument is not readily available, the convergence theory often relies on clever tricks that only work for specific stencils and/or domain types. The classical paper by Kuttler [29] develops a fairly general convergence theory for this class of methods for the Laplace operator. A list of related work can also be found in Section 9 in the review paper by Kuttler and Sigillito [30]. Because of the simplicity of the discretization, finite difference methods are also used for higher order elliptic operators, see [33, 34].

For the finite element method, the survey paper by Boffi [11] elegantly describes the essentials of the convergence theory of FEM for eigenvalue problems. The paper deals mainly with symmetric problems, with some comments and references to the non-symmetric case. It also contains many references to relevant work up to around 2010. For more on non-self adjoint problems, see Bramble and Osborn [12]. For problems that contain eigenfunctions with less regularity, e.g., the L-shpaed domain, an adaptive mesh refinement approach may be preferred, see Dai, Xu and Zhou [17] and also Carstensen and Gedicke [14], where their methods also provide guaranteed lower bounds for the eigenvalues concerned. For implementation, FreeFem++ ([24],

available at `www.freefem.org`) is a domain-specific language based on C++ that makes the tasks of gridding and stiffness matrix calculations much easier. The documentation available on the web site contains examples on PDE eigenvalue problems. For more on FEM for eigenvalue problems, we recommend the very readable and comprehensive treatise by Babuška and Osborn [5].

There are other methods for solving PDE eigenvalue problems than the ones examined in this manuscript. There is the Method of Particular Solutions, which assumes that the eigenfunction can be written as a linear combination of eigenfunctions with certain symmetries, and the coefficients are chosen so that the boundary conditions are satisfied. This method was used by Fox, Henrici and Moler to calculate the eigenfunctions of the L-shaped domain in [19]. The method was subsequently improved by [9]. A different possibility is to use a boundary integral method, in which the eigenvalue problem is reformulated in terms of unknown traces (function values or normal derivatives) along the boundary. This has the advantage of reducing the number of unknowns dramatically and can be beneficial when searching for large eigenvalues, where the highly oscillatory eigenfunctions would require a very fine mesh in the interior of the domain. For an easy-to-understand algorithmic description of the method, see Bäcker [8]. More recent work that considers a problem with both Dirichlet and Neumann boundaries can be found in [1].

The solution of eigenvalue problems for dense matrices is now a fairly mature technology, with highly efficient implementations available in libraries such as LAPACK ([3], `www.netlib.org/lapack`) and its parallel version, ScaLAPACK ([10], `www.netlib.org/scalapack`). Nonetheless, new shift-and-deflate strategies have been proposed that lead to even faster convergence, see [28]. For sparse matrices, considerations such as eigenvalue distribution and sparsity preservation become important. In addition to the Lanczos method, another popular method is the Jacobi–Davidson method, which is described in [39]. Some numerical experiments suggest that Jacobi–Davidson does a better job in approximating interior eigenvalues than methods such as Lanczos; for more details, see the relevant references contained in [39]. We also recommend the excellent book by Saad [37], which contains very thorough discussions of the theory and algorithms related to large, sparse matrix eigenvalue problems.

Finally, we mention the book by Chatelin [15], which develops the theory and numerical methods for spectral approximation not only for differential operators, but also for other types of linear operators, e.g., integral operators.

Acknowledgment. The author would like to thank Martin J. Gander, Walter Gander, Ronald D. Haynes and Nilima Nigam for their careful reading of the manuscript and for their helpful suggestions, which significantly improved the quality of these notes. We also thank Catherine Bénéteau, Alexandre Girouard, Dmitry Khavinson, Javad Mashreghi and Thomas J. Ransford for organizing the 2016 CRM Summer School on Spectral Theory and Applications, at which the author benefited from many insightful and stimulating discussions. Finally, we thank the anonymous referees for their many suggestions that led to significant improvements in the manuscript.

7. Exercises

This section contains three problem sets handed out during the 2016 CRM Summer School on Spectral Theory and Applications. They are computational in nature, with the aim of helping participants gain insight into the theory by implementing the numerical methods themselves and experimenting with them. Some additional code was provided to the participants, in particular the finite element modules; those can be downloaded at `http://www.math.hkbu.edu.hk/~felix_kwok/crm/`.

7.1. **Problem Set #1: Finite Difference Methods.**

Exercise 1: Grid Setup and Visualization

The goal of this exercise is to learn how to set up a rectangular grid in MATLAB, as well as visualize functions defined on a grid.

1. Open a new script file in the editor and type the following commands:

   ```
   a = 0; b = 1;
   N = 10;         % Number of mesh widths in the grid
   h = (b-a)/N;    % Mesh size
   t = (a:h:b)'
   ```

   The last command generates a vector of $N+1$ points from `a` to `b` with increment `h`. Save the script and run it, either by typing its name in the command window, or by clicking the 'Run' button. We will change or include additional commands in this file in the subsequent exercises.
2. Generate a grid of x-values by creating an $(N+1) \times (N+1)$ array with $N+1$ copies of the column vector `t` using the command

   ```
   xx = repmat(t, 1, N+1)
   ```
3. Generate a grid of y-values by creating an $(N+1) \times (N+1)$ array with $N+1$ copies of the row vector `t'`, the transpose of `t`.
4. To visualize the function $f(x, y) = xy^2$, we first generate a **grid function** `u` by evaluating f at our grid points `xx` and `yy`:

   ```
   f = @(x,y) x.*(y.^2);
   u = f(xx,yy);
   ```

   Now we can plot the function using the `surf` or the `mesh` command:

   ```
   surf(xx,yy,u)
   xlabel('x')
   ylabel('y')
   ```

   The last two commands label the axes. The `mesh` command has an identical syntax to `surf`, but plots the surface a bit differently.
5. To add more resolution to the plot, change `N` in part 1 to `N = 50` and rerun your script.

Exercise 2: Laplacian on a Square

We now set up the finite difference matrix for the square grid in Exercise 1.

1. First, we number the interior points using the commands

   ```
   G = zeros(size(xx));
   G(xx > a & xx < b & yy > a & yy < b) = (1:(N-1)^2);
   ```

   Explain what this does by displaying the variable `G` (it may be easier to visualize for $N = 10$).

2. Create an empty $(N-1)^2 \times (N-1)^2$ sparse matrix A using the `sparse` command. Next, fill in the entries by completing the following loop:

```
for i=1:size(G,1),
    for j=1:size(G,2),
        if (G(i,j)>0),
            A(G(i,j),G(i,j)) = 4/h^2;
            if (G(i+1,j)>0), A(G(i,j),G(i+1,j)) = -1/h^2; end;
            ... COMPLETE HERE ...
        end;
    end;
end;
```

3. Compute the five smallest eigenvalues and their associated eigenvectors using the command `eigs`. Do not forget the 's' at the end – `eig` is a different command! If `v` is an eigenvector of `A`, we can visualize it as an eigenfunction on the unit square by running

```
U = G;
U(G>0) = v;
surf(xx,yy,U);
```

4. Compute the **smallest** eigenvalue for different mesh sizes by letting $N = 8$, 16, 32, 64, 128. Plot the difference between these approximations and the exact value $\lambda = 2\pi^2$ in a log-log plot, using the command `loglog`. Also plot on the same graph the curves $y(N) = N^{-\alpha}$ for $\alpha = 1, 2, 3$. At what rate does the smallest eigenvalue converge to the exact value?

Exercise 3: L-shaped Domain

The MATLAB commands `numgrid` and `delsq` provide a quick way of setting up the discrete Laplacian matrix. To set up the Laplacian for a square, use the pair of commands

```
G = numgrid('S',N+1)
A = delsq(G);
```

1. Check that `numgrid` and `delsq` give the same grid and matrix as the ones generated in Exercise 2, provided that the appropriate mesh size h is used.
2. Generate a grid for the L-shaped domain shown in the lecture slides, i.e., the region $[-1, 1] \times [-1, 1]$ with the second quadrant removed. Calculate the smallest eigenvalue and the associated eigenfunction for $N = 16, 32, 64, \ldots, 512$.
3. If we have a slowly converging sequence, such as our sequence of eigenvalues, we can transform it into a new, faster-converging sequence with the same limit using the ϵ-algorithm by P. Wynn (cf. [21, §5.2.4]). The function `EpsilonAlgorithm` provided returns the last element of this new sequence. Apply this function to the sequence of eigenvalues obtained in part 2. We will consider the answer to be our best approximation to the exact eigenvalue.
4. Study the convergence of the first eigenvalue as a function of $h = 1/N$. Does the error behave like $O(h^2)$? What is the actual convergence rate?
5. Repeat the above computation for the next four smallest eigenvalues. Why do some eigenvalues converge faster than others?

Exercise 4: Neumann Conditions

In this exercise, we will solve numerically for the eigenvalues of the Laplacian of

the unit square with Neumann boundary conditions

$$-\Delta u = \lambda u \quad \text{on } \Omega = (0,1)^2, \quad \frac{\partial u}{\partial n} = 0 \quad \text{on } \partial\Omega.$$

Recall that the finite volume method gives the *generalized* eigenvalue problem

$$A\mathbf{u} = \lambda B\mathbf{u},$$

where A has the same structure as the finite difference matrix at interior points, and B is a diagonal matrix.

1. Modify your code to set up the numbered grid `G`, the stiffness matrix `A` and the mass matrix `B`. Points to note:
   - You may set up the matrix A either from scratch like in Exercise 2, or by using `delsq` and dividing appropriate parts of the matrix by 2. Either way, remember to scale the entries correctly along boundary nodes!
   - Be careful with the **size** of these matrices: nodes on the boundary are now part of the degrees of freedom in $\mathbf{u}$.

   For $N = 16$, the first five eigenvalues should be 0.0000, 9.8379, 9.8379, 19.6759, 38.9737.
2. Notice that the first computed eigenvalue is always zero up to machine precision. Can you prove it?
3. Verify numerically that the error of the computed eigenvalues is $O(h^2)$. Can you prove this analytically?

7.2. **Problem Set #2: Finite Element Methods.**

Exercise 1: Getting to know a finite element code

In this exercise, you will familiarize yourself with the different parts of a finite element code for solving eigenvalue problems. The following routines are included in the sample code:

NewMesh.m: sets up the data structures for the initial triangular mesh;
PlotMesh.m: displays the mesh stored in the data structures
RefineMesh.m: refines a given mesh by a factor of two;
SmoothMesh.m: moves the nodes of a mesh to get a more uniform triangulation;
ComputeElementStiffnessMatrix.m: computes the element stiffness matrix for a given triangle;
ComputeElementMassMatrix.m: computes the element mass matrix for a given triangle;
FEeig.m: computes the assembled stiffness and mass matrices for a given mesh;
FEeig_fast.m: same as `FEeig.m`, but optimized to exploit the built-in MATLAB routines for manipulating sparse matrices.
PlotSolution.m: displays the finite element function $u(\mathbf{x}) = \sum_i u_i \varphi_i(\mathbf{x})$ given its vector of degrees of freedom $\mathbf{u} = (u_i)_{i=1}^N$.

The following commands show how to run the code from start to finish for the unit square. To run all of them sequentially, save them into an M-file (say `eig_example.m`), then click the 'Run' button or type the name of the file (without the `.m`) on the command line.

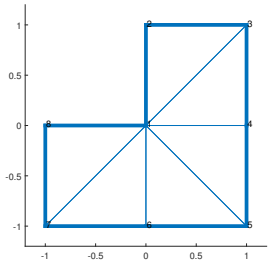

FIGURE 19.

1. Change the argument to `NewMesh` to see what other domains are available.
2. Refine the mesh for the unit square until there are at least 9 degrees of freedom. Compute the five smallest eigenvalues of $-\Delta u = \lambda u$, $u|_{\partial\Omega} = 0$, and display their corresponding eigenfunctions. Do the eigenfunctions look the way you expect them to?
3. Continue refining the grid and calculate the first five eigenvalues for each level of refinement. Plot the error as a function of the mesh parameter h. Does the error behave like $O(h^2)$? (**Hint:** If the code becomes too slow, replace `FEeig` by `FEeig_fast` in your code.)

Exercise 2: L-shaped domain

In this exercise, you will create an initial mesh for the L-shaped domain shown in Figure 19 and study the convergence of its eigenfunctions.

1. Create the array `N` containing in each row i the x and y coordinates of the ith node. The nodes can appear in any order.
2. Create the array `T` containing a list of triangles defined by the nodes. Each row of `T`, say `T(i,:)`, contains six entries according to the format

   ```
   [v1, v2, v3, e1, e2, e3]
   ```

   where `v1`, `v2` and `v3` are the three nodes of the ith triangular element **in counterclockwise order**. The fourth entry `e1` is 1 if the edge (v_1, v_2) is part of the physical boundary, and zero otherwise. The fifth and sixth entries are similar, except they describe the edges (v_2, v_3) and (v_3, v_1). For example, one of the triangles in the mesh above is described by the row `[1 3 2 0 1 1]`. The triangles can appear in any order.
3. Verify whether your mesh is set up correctly by visualizing it using `PlotMesh`. Once everything is correct, add this new mesh to your copy of `NewMesh.m`.
4. Study the convergence of the first few eigenvalues of $-\Delta u = 0$, $u|_{\partial\Omega} = 0$ for the L-shaped domain under mesh refinement. Do you observe the same phenomenon as in the finite difference case?
5. Suppose we want to study the eigenvalues for the Neumann problem, i.e., we want $\nabla u \cdot n|_{\partial\Omega} = 0$. How would you modify your mesh generation code to handle this? Compute the first few eigenvalues and eigenfunctions for the Neumann problem for different shapes.

Exercise 3: Robin boundary conditions

In this exercise, we consider the eigenvalue problem with Robin boundary conditions

$$-\Delta u = \lambda u \quad \text{on } \Omega, \qquad \nabla u \cdot n + pu = 0 \quad \text{on } \partial\Omega,$$

where $p > 0$ is a constant for simplicity. Recall that in this case, the energy bilinear form contains the extra boundary integral $p \int_{\partial\Omega} uv \, dS(x)$. We will compute the contribution of this term edge by edge along the physical boundary.

1. Given an edge $\mathcal{E}$ defined by the nodes (x_1, y_1), (x_2, y_2), calculate (with pencil and paper) the 2×2 **edge** mass matrix corresponding to the integrals $\int_{\mathcal{E}} \varphi_i \varphi_j \, dS(x)$, $i, j = 1, 2$.
2. Using the function `ComputeElementMassMatrix` as a model, write a function `ComputeEdgeMassMatrix` that implements your calculation in part 1.
3. Modify the assembly routine `FEeig` to incorporate the Robin boundary term. Save your result under a different name, e.g., `FEeig_Robin.m`. Check that your results are correct by calculating the first eigenvalue and eigenfunction for the unit square for $p = 1$. As you refine the grid, this value should approach $\lambda_1 \approx 3.4141\ldots$

7.3. **Problem Set #3: Matrix Eigenvalue Problems.**

Exercise 1: Sparse vs. Dense Matrices

In MATLAB, a matrix can be stored in *dense* or *sparse* format.

- A **dense** matrix is stored in a contiguous block of memory for n^2 real numbers, and zero entries are stored explicitly as the real number '0'.
- A **sparse** matrix only allocates enough memory for storing the non-zero entries in the matrix, plus some supporting data structures for locating these non-zero entries within the matrix. Zeros are not explicitly stored.

MATLAB provides two commands, `eig` and `eigs`, for computing the eigenvalues of dense and sparse matrices respectively. You have already used `eigs` in previous problem sets for calculating a few eigenvalues of sparse matrices arising from PDE discretizations. The aim of this exercise is to compare the two routines.

1. Generate the matrix of a discrete Laplacian using `numgrid` and `delsq` for $N = 4, 8, 16, 32, 64$. Use `eig` to compute **all** eigenvalues of the matrix. Note how much time the computation takes by enclosing your `eig` command within a tic-toc pair:

   ```
   tic; lambdas = eig(A); toc
   ```

   (Depending on the version, you may need to convert A to dense format first by calling `eig(full(A))`.) How does the running time behave as the size of the matrix increases?
2. Again for $N = 4, 8, 16, 32, 64$, use `eigs` to calculate (i) the 5 smallest eigenvalues, and (ii) the $m/2$ smallest eigenvalues, where m is the size of the matrix. Round to the nearest integer if m is odd. Note again the running time for each case.
3. Under what circumstances is `eigs` more efficient than `eig`, and vice versa?

Exercise 2: Power Iteration

1. Write a function `PowerMethod` that implements the power method. Use the following header:

```
function [lambda, y] = PowerMethod(A, y0)
% POWERMETHOD calculates the largest eigenvalue in magnitude using the
% power method
%  [lambda, y]=PowerMethod(A,y0) uses the power method to approximate
%  the largest eigenvalue lambda of A, using y0 as the initial vector.
%  The return value y is an approximate eigenvector associated with
%  lambda.
```

You may use the function `ShiftInvert` as a reference when writing this routine.

2. Let A be a symmetric positive definite matrix, so that all its eigenvalues are positive. There are two ways of finding its smallest eigenvalue:
   - Use the inverse power method (i.e., shift-and-invert with a zero shift),
   - Apply the power method to the matrix $\gamma I - A$, where γ is an upper bound on the largest eigenvalue of A, e.g., its maximum row sum $\gamma = \|A\|_\infty$.

   Which method is faster in terms of (i) number of iterations and (ii) running time? For the running time, you need to consider the cost of solving the linear system $A\mathbf{x} = \mathbf{b}$ versus multiplying A by a vector $\mathbf{x}$. Try to construct matrices for which one method is faster than the other, and vice versa.

Exercise 3: Tridiagonal matrices

Write a function `Sturm` that calculates the kth eigenvalue of a symmetric tridiagonal matrix. Use the following header:

```
% STURM calculates the kth eigenvalue of a symmetric tridiagonal matrix
%  [lambda, y] = sturm(alpha, beta, k) calculates the kth eigenvalue lambda
%  of the symmetric tridiagonal matrix A with [beta(i-1), alpha(i), beta(i)]
%  in its i-th row. beta is assumed to be non-zero everywhere. The method
%  uses Sturm sequences to find a good enough approximation to the k-th
%  eigenvalue, then shift-and-invert is used to find a better approximation
%  as well as the corresponding eigenvector y.
```

To write such a function, proceed as follows:

1. First, use Gershgorin's theorem to estimate the interval $[y, z]$ containing all the eigenvalues:

   Gershgorin's Theorem. If λ is an eigenvalue of a matrix A, then there exists an index i such that $|\lambda - a_{ii}| \leq \sum_{j \neq i} |a_{ij}|$.

   In other words, all the eigenvalues must be contained in intervals of the form $[a_{ii} - \rho_i, a_{ii} + \rho_i]$, where $\rho_i = \sum_{j \neq i} |a_{ij}|$. One can thus obtain y and z by taking some minima and maxima.
2. Next, bisect the interval until $\omega(y) = k-1$ and $\omega(z) = k$. To do so, you will need to evaluate $p_0(\lambda), \ldots, p_n(\lambda)$ at the midpoint $x = (y + z)/2$ using the recurrence shown on the slide, and then count the number of sign changes in the sequence.
3. Bisect the interval two more times so that the midpoint obtained is much closer to the kth eigenvalue than to any other eigenvalue.
4. Finally, call `ShiftInvert` to obtain accurate values for the eigenpair.

Test your function on the matrix

$$A = \begin{bmatrix} 2 & -1 & & & \\ -1 & 2 & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{bmatrix},$$

whose kth eigenvalue is given by $4\sin^2\left(\frac{k\pi}{2(n+1)}\right)$. As a further test, generate a symmetric matrix A (e.g. the Laplacian for an L-shaped domain), then run the following:

```
T = hess(full(A));
[lambda, y] = sturm(diag(T),diag(T,-1),7);
```

The first command calculates the tridiagonal form of A having the same eigenvalues, and the `diag` calls inside `sturm` extract the diagonal and subdiagonal of T before passing it to `sturm`, which then computes the seventh smallest eigenvalue. Compare your results with that obtained by `eig` or `eigs`.

Exercise 4: Lanczos Method

1. Derive the Lanczos method for the *generalized* eigenvalue problem

   $$A\mathbf{x} = \lambda B\mathbf{x},$$

   where A and B are symmetric and B is positive definite. Make sure your method can be expressed entirely in terms of the following operations:
   - Scalar operations;
   - Vector manipulations of the form $\mathbf{z} \leftarrow \alpha\mathbf{x} + \mathbf{y}$;
   - Dot products between two vectors;
   - The matrix vector product $A\mathbf{x}$,
   - Linear solves involving B, i.e., solve $B\mathbf{x} = \mathbf{y}$.
2. Using `myLanczos` as a model, implement the Lanczos method for the generalized eigenvalue problem. Use your routine to calculate the lowest eigenmode for the Laplacian problem discretized by the finite element method. Taking the result from `eigs` as the exact solution, plot the error of the smallest eigenvalue as a function of the number of Lanczos iterations.
3. Compare the convergence of Lanczos with that of the power method. Which method converges faster?

References

[1] Eldar Akhmetgaliyev, Oscar P. Bruno, and Nilima Nigam. A boundary integral algorithm for the Laplace Dirichlet-Neumann mixed eigenvalue problem. *Journal of Computational Physics*, 298:1 – 28, 2015.

[2] Grégoire Allaire and Sidi Mahmoud Kaber. *Numerical Linear Algebra*. Springer, New York, 2008.

[3] Edward Anderson, Zhaojun Bai, Christian Bischof, L. Susan Blackford, James Demmel, Jack Dongarra, Jeremy Du Croz, Anne Greenbaum, Sven Hammarling, Alan McKenney, and Danny C. Sorensen. *LAPACK Users' Guide*. SIAM, 1999.

[4] Jared L Aurentz, Thomas Mach, Raf Vandebril, and David S Watkins. Fast and backward stable computation of roots of polynomials. *SIAM Journal on Matrix Analysis and Applications*, 36(3):942–973, 2015.

[5] Ivo Babuška and John Osborn. Eigenvalue problems. *Handbook of numerical analysis*, II:641–787, 1991.
[6] Ivo Babuška and Juhani Pitkäranta. The plate paradox for hard and soft simple support. *SIAM Journal on Mathematical Analysis*, 21(3):551–576, 1990.
[7] I. Babuška. Stability of the domain under perturbation of the boundary in fundamental problems in the theory of partial differential equations principally in connection with the theory of elasticity, parts I and II. *Czechoslovak Math. J*, 11:75–105, 1961.
[8] Arnd Bäcker. Numerical aspects of eigenvalue and eigenfunction computations for chaotic quantum systems. In *The mathematical aspects of quantum maps*, pages 91–144. Springer, 2003.
[9] Timo Betcke and Lloyd N. Trefethen. Reviving the method of particular solutions. *SIAM Review*, 47(3):469–491, 2005.
[10] L. Susan Blackford, Jaeyoung Choi, Andy Cleary, Eduardo D'Azevedo, James Demmel, Inderjit Dhillon, Jack Dongarra, Sven Hammarling, Greg Henry, Antoine Petitet, K. Stanley, David W. Walker, and R. Clint Whaley. *ScaLAPACK Users' Guide*. SIAM, 1997.
[11] Daniele Boffi. Finite element approximation of eigenvalue problems. *Acta Numerica*, (2010):1–120, 2010.
[12] James H. Bramble and J.E. Osborn. Rate of convergence estimates for nonselfadjoint eigenvalue approximations. *Mathematics of Computation*, 27(123):525–549, 1973.
[13] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer Science+Business Media, New York, third edition, 2008.
[14] C. Carstensen and J. Gedicke. Guaranteed lower bounds for eigenvalues. *Math. Comp.*, 83(290):2605–2629, 2014.
[15] Françoise Chatelin. *Spectral approximation of linear operators*. SIAM, 2011.
[16] Ernst Florens Friedrich Chladni. Neue Beiträge zur Akustik. In *Entdeckungen über die Theorie des Klanges*. Leipzig, 1821.
[17] Xiaoying Dai, Jinchao Xu, and Aihui Zhou. Convergence and optimal complexity of adaptive finite element eigenvalue computations. *Numerische Mathematik*, 110(3):313–355, 2008.
[18] Lawrence C. Evans. *Partial Differential Equations*. AMS, Providence, RI, 1998.
[19] L. Fox, P. Henrici, and C. Moler. Approximations and bounds for eigenvalues of elliptic operators. *SIAM Journal on Numerical Analysis*, 4(1):89–102, 1967.
[20] Martin J. Gander and Felix Kwok. Chladni figures and the Tacoma bridge: motivating PDE eigenvalue problems via vibrating plates. *SIAM Review*, 54(3):573–596, 2012.
[21] Walter Gander, Martin J. Gander, and Felix Kwok. *Scientific Computing: an Introduction using Maple and MATLAB*, volume 11 of *Texts in Computational Science and Engineering*. Springer International Publishing, Switzerland, 2014.
[22] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Prentice-Hall, Englewood Cliffs, NJ, 1963.
[23] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, third edition, 1996.
[24] Frédéric Hecht. New development in FreeFem++. *Journal of Numerical Mathematics*, 20(3-4):251–266, 2012.
[25] Bert Hubbard. Bounds for eigenvalues of the free and fixed membrane by finite difference methods. *Pacific Journal of Mathematics*, 11(2):559–590, 1961.
[26] Bert Hubbard. Eigenvalues of the non-homogeneous rectangular membrane by finite difference methods. *Archive for Rational Mechanics and Analysis*, 9(1):121–133, 1962.
[27] Claes Johnson. *Numerical solution of partial differential equations by the finite element method*. Courier Corporation, 2012.
[28] Bo Kågström and Daniel Kressner. Multishift variants of the QZ algorithm with aggressive early deflation. *SIAM Journal on Matrix Analysis and Applications*, 29(1):199–227, 2006.
[29] James R Kuttler. Finite difference approximations for eigenvalues of uniformly elliptic operators. *SIAM Journal on Numerical Analysis*, 7(2):206–232, 1970.
[30] J.R. Kuttler and V.G. Sigillito. Eigenvalues of the Laplacian in two dimensions. *Siam Review*, 26(2):163–193, 1984.
[31] Richard S. Laugesen. Spectral Theory of Partial Differential Equations. Available at `https://wiki.cites.illinois.edu/wiki/display/MATH595STP/Math+595+STP`, April 2017.
[32] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, 1998.

[33] V. G. Prikazchikov. The finite difference eigenvalue problem for fourth-order elliptic operator. *U.S.S.R. Comput. Maths Math. Phys.*, 17:89–99, 1978.
[34] V. G. Prikazchikov and A. N. Khimich. The eigenvalue difference problem for the fourth order elliptic operator with mixed boundary conditions. *U.S.S.R. Comput. Maths Math. Phys.*, 25:137–144, 1985.
[35] P.-A. Raviart and J. M. Thomas. *Introduction à l'Analyse Numérique des Équations aux Dérivées Partielles*. Collection Mathématiques Appliquées pour la Maîtrise. Masson, Paris, 1983.
[36] Walther Ritz. Theorie der Transversalschwingungen einer quadratischen Platte mit freien Rändern. *Annalen der Physik*, 18(4):737–807, 1909.
[37] Youcef Saad. *Numerical methods for large eigenvalue problems*. SIAM, second edition, 2011.
[38] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, 2nd edition, 2003.
[39] Gerard L.G. Sleijpen and Henk A. Van der Vorst. A Jacobi–Davidson iteration method for linear eigenvalue problems. *SIAM Review*, 42(2):267–293, 2000.
[40] Walter A. Strauss. *Partial Differential Equations: an Introduction*. John Wiley & Sons, Inc., Hoboken, NJ, 2008.
[41] Andrea Toselli and Olof B. Widlund. *Domain Decomposition Methods – Algorithms and Theory*. Springer, Berlin Heidelberg, 2005.
[42] H.F. Weinberger. Upper and lower bounds for eigenvalues by finite difference methods. *Communications on Pure and Applied Mathematics*, 9(3):613–623, 1956.
[43] H.F. Weinberger. Lower bounds for higher eigenvalues by finite difference methods. *Pacific J. Math*, 8(2):339–368, 1958.
[44] James H Wilkinson. The evaluation of the zeros of ill-conditioned polynomials. part I. *Numerische Mathematik*, 1(1):150–166, 1959.
[45] J.H. Wilkinson. *The Algebraic Eigenvalue Problem*. Clarendon Press, Oxford, England, 1965.