PDEs: Eigenvalue Problems

Aleksandar Ivanov

March 31, 2021

1 Problem Statements

Problem 1

Find a couple of the lowest lying eigenvalues and eigenmodes of a square membrane with variable density as shown in fig. 1, by translating the problem into a system of difference equations.

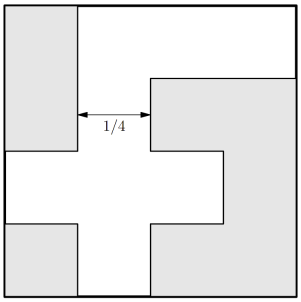

Figure 1: Shape of the membrane and its density distribution. The darker parts represent a higher density.

The darker region in the figure represents a higher density. What would happen if it was smaller instead? Look at the spectrum as a function of different values of this density.

Try out different methods and compare their convergence, speed and accuracy on the normal square membrane.

Problem 2

Estimate the eigenvalues of the Laplacian ∇^2 on the semicircular domain with Dirichlet boundary conditions.

2 Mathematical Setup

As suggested in the problem statement, we will be solving this problem by discretizing the Laplacian operator. For the first problem we're solving the equation

$$\nabla^2 u + \frac{\omega^2 d}{\gamma}\rho(x,y)u = 0, \tag{1}$$

where $\omega^2 d/\gamma$ is a constant that effectively gives the eigenvalue and $\rho(x,y)$ is a function representing the density of the membrane at different points. To discretize this we will organize the solution u into a vector in the following way

$$u^{\text{vec}} = (u_{00}, u_{01}, \cdots, u_{10}, \cdots) \tag{2}$$

Since we're working on a square domain, discretization of the derivatives is straightforward, and it generates a symmetric matrix like the following example for four points

$$A_{N=4} = \begin{bmatrix} 4 & -1 & 0 & 0 & -1 & 0 & \cdots \\ -1 & 4 & -1 & 0 & 0 & -1 & \cdots \\ 0 & -1 & 4 & -1 & 0 & 0 & \cdots \\ 0 & 0 & -1 & 4 & 0 & 0 & \cdots \\ -1 & 0 & 0 & 0 & 4 & -1 & \cdots \\ 0 & -1 & 0 & 0 & -1 & 4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3}$$

In other words, the matrix $A \in \mathbb{R}^{N^2 \times N^2}$ is made up of tridiagonal blocks of size $N \times N$ on its main diagonal with an additional line of values on its $N+1$-st diagonal in both directions.

Here we have a choice to make on which side to include $\rho(x,y)$. If we include it on the matrix side, we break the symmetry of the matrix and lose out on both efficiency and ease of use. This is why

we instead choose to solve a generalized eigenvalue problem with $\rho(x, y)$ playing the role of the weight matrix B

$$Au = \lambda Bu. \tag{4}$$

The B in this equation is a diagonal matrix and is constructed from ρ with the prescription

$$\rho(x_i, y_j) = B_{iN+j,iN+j}. \tag{5}$$

The different definitions for the eigenvalues are related by

$$N^2\lambda = \frac{\omega^2 d}{\gamma} = k^2 \tag{6}$$

3 Numerical Methods

In this work we will use and compare four different numerical methods for solving this diagonalization problem. The simplest one is the Power Iteration method. This method take an initial eigenvector guess z_0 and generates the sequence

$$z_{k+1} = \frac{Az_k}{||Az_k||} \tag{7}$$

of successive applications of the matrix. If we decompose the vector z_0 into the eigenvectors of A, we can see that the component with the highest eigenvalue gets boosted the most, and since we're normalizing in the end this will eventually converge to the eigenvector with the highest eigenvalue. The convergence, however, is only in the direction of the eigenvector, so we can't use a normal stopping condition. Instead, we define the Rayleigh quotient

$$\rho(x, A) = \frac{x^T Ax}{x^T x}, \tag{8}$$

which is an estimate for the eigenvalue, and we use the stopping condition

$$||Az_k - \rho(z_k, A)z_k|| < \epsilon, \tag{9}$$

which doesn't have to worry about consecutive approximations having different directions.

This procedure to get the largest eigenvalue can be extended to a procedure which computes all eigenvalues by using some reduction of the matrix. For symmetric matrices a particularly nice choice is Hotelling reduction. After having computed the eigenpair (λ_1, x_1), we can define the matrix

$$\tilde{A} = A - \lambda_1 x_1^T x_1, \tag{10}$$

which shares all its eigenvalues with A, except that the largest one has now been replaced by a 0. From here we compute the second largest eigenvalues and continue until we have as many eigenvalues as we want.

We will also be using the related method of Inverse Iteration, which is just power iteration applied to the inverse matrix A^{-1}. The benefit of this method is that it finds the eigenvalues of the original matrix A from the smallest to the largest instead of the other way around, as is the case of power iteration. This is more efficient for our type of problem because the discretization means that the only correct part of the spectrum is the low-lying part.

Inverse iteration can also be modified to find a specific eigenvalue by iterating with the matrix $(A-\sigma I)^{-1}$, where σ is our guess for the eigenvalue.

Another method that we will look at is Jacobi iteration, which uses the idea of multiplying by rotation matrices to, one by one, set to 0 all off-diagonal elements. The method rotates by the Givens' matrix

$$R_{ij}(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{11}$$

where the i, j signifies the row and column which we are rotating about. The elements of $R_{ij}(\theta)$ can be calculated and for maximum stability they are given by

$$\tau = \frac{A_{ii} - A_{jj}}{2A_{ij}}, \qquad t = \frac{\text{sign}(\tau)}{|\tau| + \sqrt{1+\tau^2}}, \tag{12}$$

$$\cos\theta = \frac{1}{\sqrt{1+t^2}} \qquad \sin\theta = t\cos\theta, \tag{13}$$

whenever $A_{ii} \neq A_{jj}$ and by $\theta = \pi/4$ otherwise. One iteration of Jacobi is a pass through the whole matrix with these Givens rotations, after which we would, in theory, be finished. But in practice this is not the case and the procedure has to be completed at least a few times.

In our case we will use the stopping condition

$$\text{off}(A) < \epsilon, \tag{14}$$

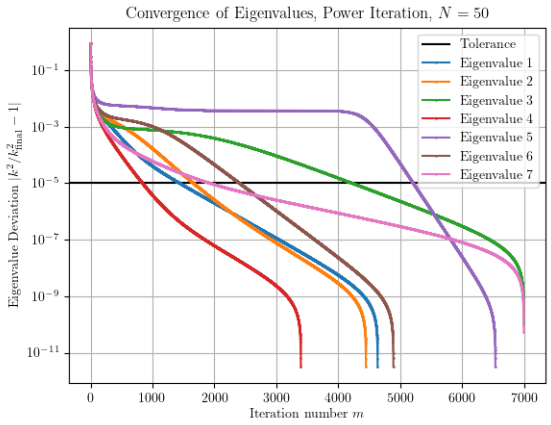

Figure 2: Convergence of the first seven eigenvalues using the power iteration method. The tolerance from the stopping condition is shown in black.

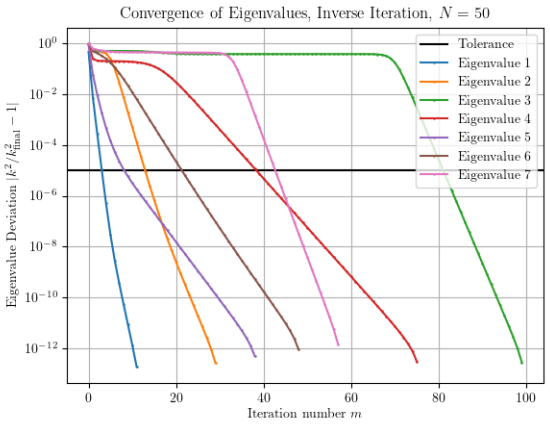

Figure 3: Convergence of the first seven eigenvalues using the inverse iteration method. The tolerance from the stopping condition is shown in black.

where the offset is defined as

$$\text{off}(A) = \sqrt{\sum_{i \neq j} |A_{ij}|^2}. \tag{15}$$

The final method we will examine is the built-in `scipy.linalg.eigh` method for eigenvalue problems of symmetric matrices. Since it is built-in, it would presumably be much too complicated to describe its inner workings, however we do know that this method is just a wrapper for the `ARPACK` library. This means that we can expect it to perform the best of the four.

4 Results

We are at first interested in how these methods converge. We will test this on the homogeneous membrane. For power iteration and inverse iteration the convergence criterion is the same, so we can compare these two. The iterations of the Jacobi method have a slightly different meaning and are thus not directly comparable, but they will still be stated.

Figure 2 shows the deviation of the eigenvalue from its final calculated value as a function of the number of iterations. The tolerance of the stopping condition is also shown as the horizontal black line. The generic trend is that the eigenvalues are determined to a very high precision and that the precision gained increases with the number of iterations. For the higher eigenvalues there is initially a plateau of improvement that then yields to the consecutive decrease. We can also see some variance in how many iterations are necessary to reach the desired tolerance, which is an effect of the random choice of z_0.

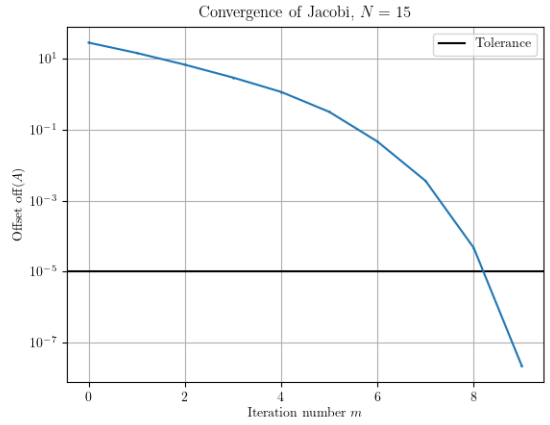

Figure 4: Convergence of the offset when using the Jacobi iteration method. The tolerance from the stopping condition is shown in black.

A similar plot for the inverse iteration method is shown in fig. 3. There we see that even though the

logic of the two algorithms is the same, inverse iteration converges much faster, in the sense of number of iterations, for our particular problem. The qualitative behavior of the convergence is the same for both methods, with the plateau again being noticeable for larger eigenvalues.

For Jacobi iteration we measure the convergence with the help of the offset. The behavior is shown in fig. 4. We see that it goes down faster over time. The main drawback of the Jacobi method, though, is that it is very slow, so that the plot could only be generated with $N = 15$.

For the built-in method we have no access to the intermediate stages of computation, so we can't really say what its convergence properties are.

We already mentioned that the Jacobi method is slow, so next lets look at the speed of convergence of these algorithms.

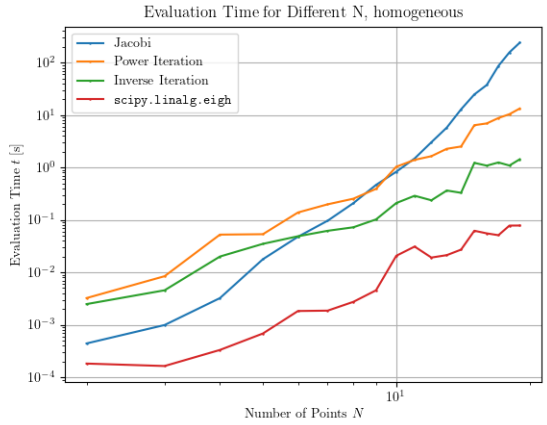

Figure 5: Evaluation times of the different methods as a function of the number of points N shown in a log-log scale.

Figure 5 shows the evaluation times for the different methods as a function of the number of points N, in a log-log scale. As we can see, the Jacobi algorithm scales much worse than the other three, whose scaling is similar. As expected, the built-in function is the fastest of the four. Furthermore, there is a difference between power and inverse iteration, with inverse iteration being faster, but not as much as one would expect from the previous comparison of the convergence. The upfront cost of inverting the matrix to prepare it for inverse iteration is not accounted for in this plot.

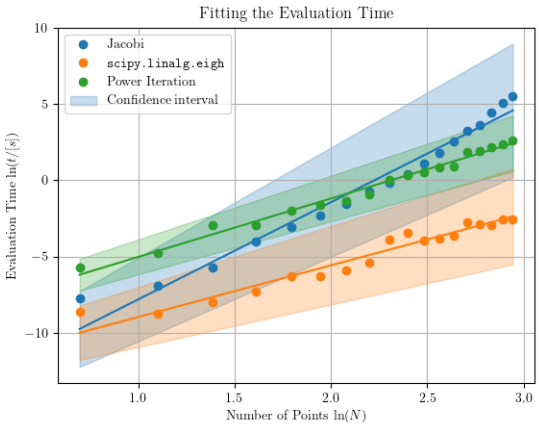

Figure 6: Power law fit of the evaluation times of the different methods as a function of the number of points N.

We can also try to get the asymptotic behavior of these evaluation times, however we can't expect too much accuracy in this regard since our number of points is very limited exactly because of the inefficiency of computing with larger N that this. Figure 6 shows the results of such a power law fit. We see that our fitting error is large, but we can confidently say that Jacobi scales with a power about twice as large as the rest. The numerical results of the fit are

$$\begin{aligned} \text{Jacobi}: &\qquad t \sim N^{\approx 6.1} \\ \text{Others}: &\qquad t \sim N^{\approx 3.3}. \end{aligned} \tag{16}$$

All times presented here were for the calculation of all of the eigenvalues. This is another point that makes Jacobi weaker; it's harder to modify Jacobi, to make it return only a few of the eigenvalues. This modification is very easy in the case of the other methods.

Finally, a method that converges but isn't accurate is useless to us, so we move on to testing the accuracy of these methods. For the square membrane we have can calculate the eigenvalues analytically with

$$k^{mn} = \pi\sqrt{m^2 + n^2}, \tag{17}$$

where m and n are indices that are natural numbers bigger that or equal to 1.

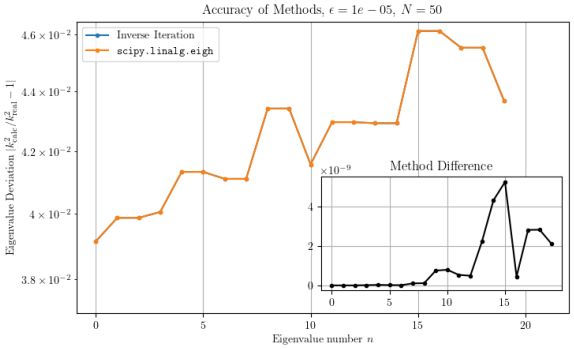

Figure 7: Difference between the real and computed values using the inverse iteration and built-in methods as a function of the number of the eigenvalue in a sorted fashion. The inset plot shows that the numerical methods agree between themselves.

Testing inverse iteration and the built-in method we compile fig. 7. It shows the difference between the analytically calculated value and the numerically calculated one as a function of the number of the eigenvalue when sorted from smallest to largest. We see that we're not as fabulously accurate as we were lead to think from the previous tests on convergence. This is not due to the methods we're using as can be seen from the inset plot, which shows that the numerical methods agree between themselves very well. It is instead due to the discretization. At the end of the day we're only using $N = 50$ points in each direction to describe this membrane, and we can't expect to have that much agreement with the continuous theoretical model. Another expected trend is that the accuracy becomes smaller as we go to larger and larger eigenvalues, which happens because large modes are more oscillatory in nature and can't be described as well by our discrete grid.

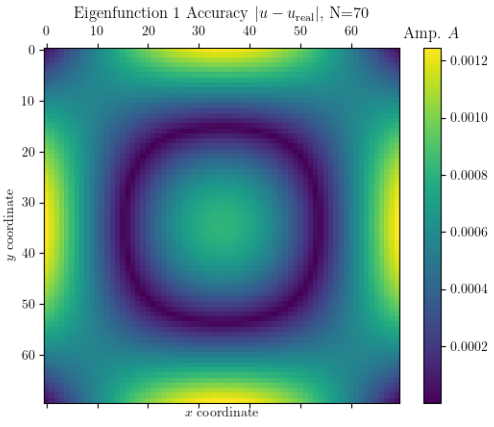

Figure 8: The error in the first eigenmode (i.e. the $m = 1$, $n = 1$ mode) using $N = 70$.

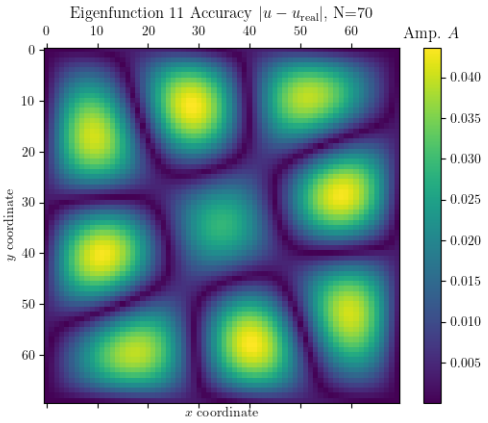

Figure 9: The error in the eleventh eigenmode (i.e. the $m = 3$, $n = 3$ mode) using $N = 70$.

Since we also have the eigenfunctions

$$u^{mn} = \sin(m\pi x)\sin(n\pi y), \tag{18}$$

where the indices are the same as for the k before, we can compare these to the calculated eigenfunctions. This comparison gets a bit complicated by the fact that some of our eigenmodes are degenerate, so that an arbitrary superposition of the degenerate modes is again a degenerate mode of the same frequency. To avoid this problem we will only compare non-degenerate modes. Here we use one low one, the first or $m = 1$, $n = 1$ mode, and one high one, the eleventh or $m = 3$, $n = 3$ mode. Figures 8 and 9 show these results. We see that the accuracy in the eigenfunctions is comparable to the accuracy in the eigenvalues. We know that the real solutions have the alternating circular extrema of the sines and that those are the places where we get the most wrong in the high eigenvalue case. In the case of the low eigenvalue the largest culprit is actually along the sides of the sides of the membrane.

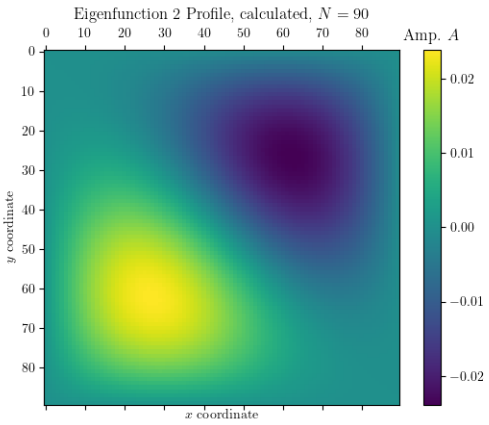

Figure 10: The second calculated eigenfunction.

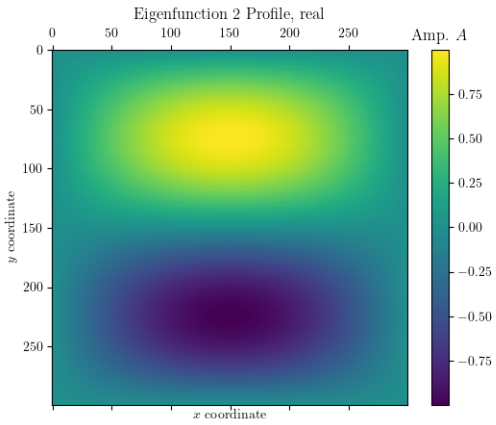

Figure 12: The second real eigenfunction.

The degenerate eigenmodes are as expected some linear superposition of the possible modes in the degenerate subspace, and usually don't align with the basis of sines. An example is the comparison between figs. 10 and 12.

The spectrum for the square membrane calculated with `scipy.linalg.eigh` is shown in fig. 11. It tells a similar story for the accuracy; the low-lying eigenvalues can be trusted and as we go higher we very quickly lose accuracy. The inset plot shows the first few eigenvalues and consequently the degeneracies seen there. It also shows that even at the tenth eigenvalue there is a visible error — again a consequence of the small number of points $N = 70$ in this case.

Having tested all of these facets of the numerical methods, we return to the original problem — generating the solutions for the membrane with a more complicated mass distribution. The only parameter here is the density of the outside region, since we're free to set the inside region's density to 1, which we will do. We will cal the density of the outside region $\tilde{\rho}$.

What follows shows a couple of the low-lying eigenfunctions that have been calculated.

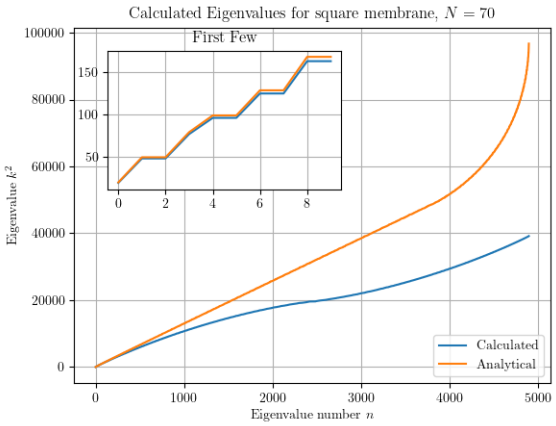

Figure 11: Spectrum for the square homogeneous membrane.

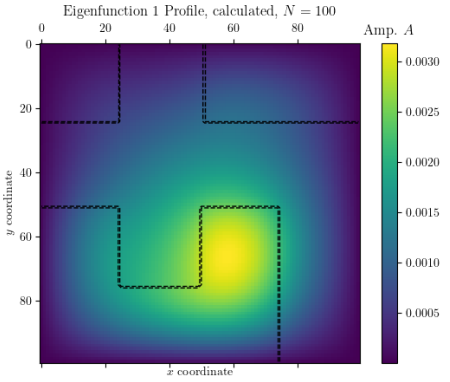

Figure 13: First eigenfunction. $\tilde{\rho} = 10$.

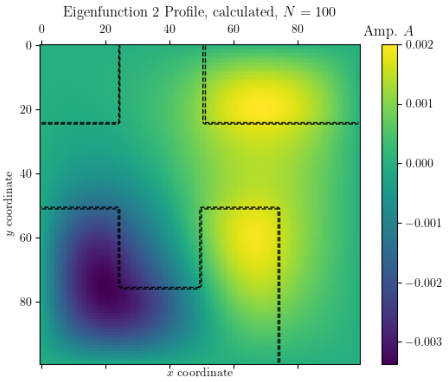

Figure 14: Second eigenfunction. $\tilde{\rho} = 10$.

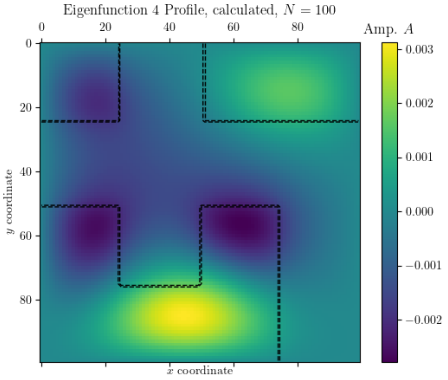

Figure 15: Forth eigenfunction. $\tilde{\rho} = 10$.

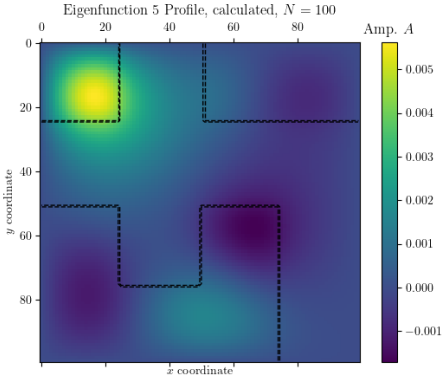

Figure 16: Fifth eigenfunction. $\tilde{\rho} = 10$.

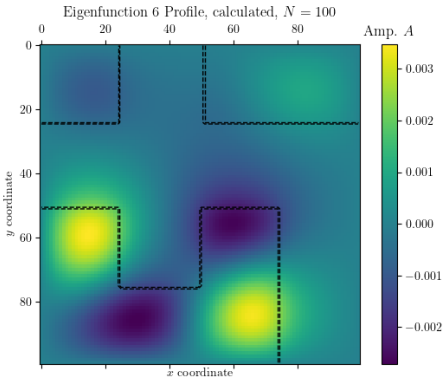

Figure 17: Sixth eigenfunction. $\tilde{\rho} = 10$.

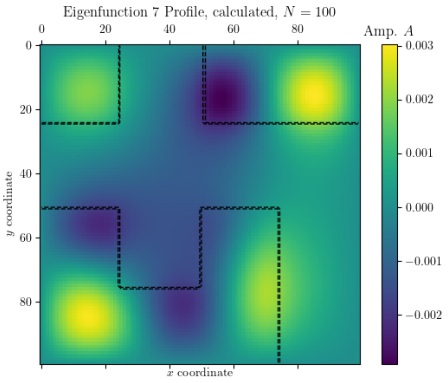

Figure 18: Seventh eigenfunction. $\tilde{\rho} = 10$.

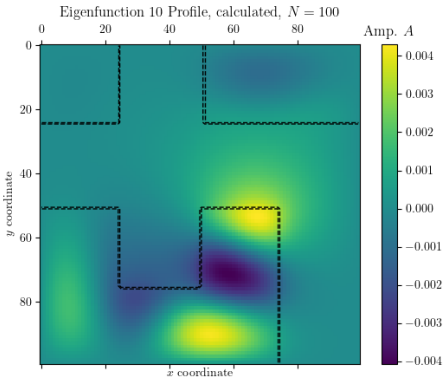

Figure 19: Tenth eigenfunction. $\tilde{\rho} = 10$.

From figs. 13 to 17 and 19 we see that, in a lot of cases, the region of higher density has larger amplitudes than the region of lower density. This is because the amplitude isn't related to how much the specific part of the membrane weighs. Rather, it's the fact that the outer region has more uniform regions while the inner region is bent and narrow; oscillations there would give rise to higher frequencies. Another interesting phenomenon that we can observe is that at the boundary between the regions we have a matching of the outer and inner solutions and this gives rise to the effect that the extrema shy away from crossing said boundary and bunching up along it.

We could say that the first eigenmode has an expected shape with one extremum, but as we go to the higher modes they quickly start to have weirder shapes.

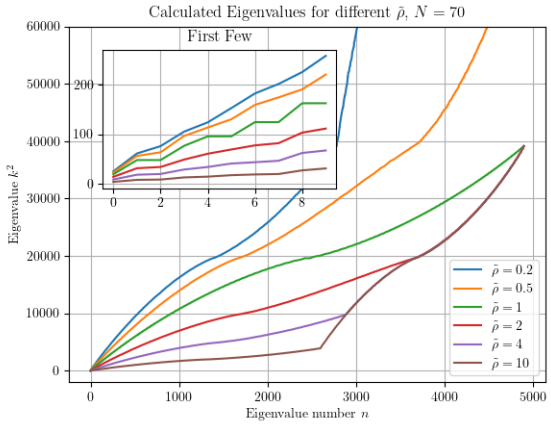

Figure 20: The spectrum for different densities of the outside region.

We are also interested in the spectrum of eigenvalues for different $\tilde{\rho}$. Figure 20 shows this spectrum as well as the most important first few eigenvalues separately. We see that the higher the density the slower the eigenvalues grow. In the second half of the figure we see this transition where the eigenvalue for different $\tilde{\rho}$ become the same, but as we learned before eigenvalues that high up are not to be trusted. From the inset, it's visible that the degeneracy of the $\tilde{\rho} = 1$ case gets broken once we have $\tilde{\rho} \neq 1$. This is to be expected since the "F" shape doesn't preserve any of the symmetries of the homogeneous square. A better look at the breaking of this degeneracy and the different eigenvalues is provided in table 1.

n	$\tilde{\rho} = 0.2$	$\tilde{\rho} = 1$	$\tilde{\rho} = 4$	$\tilde{\rho} = 10$
1	25.10	19.18	8.70	3.95
2	61.19	47.94	18.35	7.97
3	76.07	47.94	19.80	8.59
4	105.40	76.70	28.88	12.89
5	124.13	95.81	33.95	14.47
6	152.96	95.81	40.95	17.40
7	182.48	124.56	43.58	18.90
8	201.62	124.56	46.50	19.73
9	225.65	162.69	62.22	27.13
10	258.46	162.69	67.34	31.04

Table 1: First few eigenvalues for a couple of different $\tilde{\rho}$. The degeneracy of $\tilde{\rho} = 1$ is generically broken.

5 Problem 2

For the first problem our discretization gave us a symmetric matrix, the nice properties of which, we used extensively. For the second problem though, we are out of that luck. Discretizing in polar coordinates gives a matrix similar in shape to the one before, but with different values. On the diagonal now we get

$$A_{iN+j,iN+j} = 1 + \frac{N^2}{i^2}. \tag{19}$$

The second diagonals are now given by

$$A_{iN+j,iN+j\pm1} = -\frac{N^2}{i^2}. \tag{20}$$

And finally the diagonals to the sides are given by

$$A_{iN+j,iN+j\pm N} = -(1 \pm \frac{1}{2i}). \tag{21}$$

These last terms are what gives the asymmetry in the matrix.

Numerically, the above formulas with $1/i$ can't actually be achieved, so we shift the radial direction by a small amount to get rid of these. This has a large effect on some solutions, so if we are calculating all eigenvalues we have to repeat the procedure with two different shifts and check that the

eigenvalues we regard as solutions haven't changed by much. In this way, using $N = 50$, we calculate the first few eigenvalues as

$$\begin{aligned} &k_1^2 = 1.99, \quad k_2^2 = 2.67, \quad k_3^2 = 2.75, \quad k_4^2 = 4.96, \\ &k_5^2 = 4.97, \quad k_6^2 = 6.05, \quad k_7^2 = 7.02, \quad k_8^2 = 7.08. \end{aligned} \tag{22}$$

References

[1] Emil Žagar. *Problem Lastnih Vrednosti*. Slovene. [Lecture Notes].