Scientific Computing Exercise Set 3

Romy Meester (11140046), and Natasja Wezel (11027649)

I. Introduction

In biology, a membrane encompasses a microscopic double layer of lipids and proteins forming the boundary of cells or organelles. Solving the wave equation on a two-dimensional elastic material, we analyze different shapes of the membranes or drums. To be more specific, whether the membrane shape can be identified uniquely from its eigenvalue spectrum.

The aim of this research is to study the eigenmodes of drums or membranes of different shapes, and analyzing the steady state of the diffusion equation using direct methods for solving eigenvalue problems from standard python libraries.

First of all, we evaluate the eigenmodes and eigenfrequencies of membranes of different shapes. Moreover, we discretize the wave equation in order to implement the system, and construct time-dependent solutions. Besides that, we investigate the steady state concentration by discretizing the diffusion equation, and constructing a matrix equation. We explain how this equation is constructed and how the boundary conditions are taken into account for either the matrix version of the eigenvalue problem, and the direct methods.

II. Theory and Methods

A. Eigenmodes of membranes of different shapes

In order to calculate the eigenmodes and eigenfrequencies of the drum or membrane, we use boundary conditions that fix the membrame along its edge (which will be discussed in section III-A). Moreover, we consider the wave equation in two dimensions.

$$\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u \tag{1}$$

where c is the concentration, and ∇^2 the Laplacian. Also, we look for a solution in the form:

$$u(x, y, t) = v(x, y)T(t) \tag{2}$$

where the time (t) and space (x, y) dependencies are separated in two independent functions. Not every scalar function $u(x, y, t)$ can be separated this way, so we are specifically looking for a u of this form due to the nice properties. When we insert this into the wave equation and move all factors with t to the left and all factors that depend on x and y to the right, we get:

$$\frac{1}{T(t)} \frac{\partial^2 T(t)}{\partial t^2} = c^2 \frac{1}{v(x, y)} \nabla^2 v(x, y) \tag{3}$$

The left hand side only depends on t, the right hand side only on x and y. This means that both sides must equal some constant K, independent of x, y and t. Now the left and right hand side can be treated independently. We start with the left hand side:

$$\frac{\partial^2 T(t)}{\partial t^2} = K c^2 T(t) \tag{4}$$

If $K < 0$, the equation has an oscillating solution

$$T(t) = A\cos(c\lambda t) + B\sin(c\lambda t), \tag{5}$$

where $\lambda^2 = -K$. $\lambda > 0$ can be assumed without loss of generality. The solutions grows or decays exponentially when $K > 0$, so we are not interested in these solutions at the moment. If $K = 0$, the solution is a constant or a linear function of x and y, and thus it has to be $v = 0$ due to the boundary conditions.

Moreover, the right-hand side is:

$$\nabla^2 v(x, y) = K v(x, y) \tag{6}$$

which is an eigenvalue problem. The solutions to this problem give the eigenmodes v and eigenvalues λ. To find v we discretize the two-dimensional wave equation (equation 1) to obtain a set of linear equations. This system of equations is written in the form $M\nu = K\nu$, where M is a matrix, ν a vector of $v(x,y)$ at the grid points and K is a scalar constant. The matrix eigenvalue problem is solved with scipy's library methods in python, with the following boundary conditions taken into account:

$$v(x,y) = 0, \tag{7}$$

on the boundary, i.e. the boundary is fixed. Besides that, in equation 5 we take c to be 1.

After setting up the matrix, that encodes the connections of our grid, we can solve the eigenvalue problem and obtain the eigenfrequencies of our drum. We do this for drums with different shapes:

- Square with side length L
- Rectangle with sides L and 2L
- Circle with diameter L

where L describes the shape of the drum or membrane. For the circle, some points in the grid (further away than distance $R = L/2$) do not belong to the domain that can move. To implement these irregular boundary conditions, we let the vector ν contain points in a rectangular grid, in which some of them do not belong to the domain. For those, $v_k = 0$. The corresponding (k^{th}) row of the matrix will also be 0, except for a 1 at the diagonal. In the matrix eigenvalue equation, $M\nu = K\nu$, the k^{th} row will then be $0 = Kv_k$, forcing $v_k = 0$, or $K = 0$. We can ignore the eigenvalues with a value of 0. To solve the matrix equation, given in equation 8 for example a square of gridsize $N \times N$, we use different methods from the scipy library. For higher performance, the speeds of these different techniques are compared.

$$\begin{pmatrix} m_{11} & \cdots & m_{1N^2} \\ \vdots & \ddots & \vdots \\ m_{N^21} & \cdots & m_{N^2N^2} \end{pmatrix} \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{N^2} \end{pmatrix} = K \begin{pmatrix} \nu_1 \\ \vdots \\ \nu_{N^2} \end{pmatrix} \tag{8}$$

B. *Direct methods for solving steady state problems*

In the second part of this assignment, we will find the steady state of the diffusion equation. We will use a direct method, instead of the iterative methods used in the previous assignments. Consider again the two-dimensional time dependent diffusion equation:

$$\frac{\partial c}{\partial t} = D\nabla^2 c \tag{9}$$

Here, $c(x,y;t)$ is the concentration as a function of the coordinates x and y and time t. D is the diffusion constant. A matrix for the ∇^2 is constructed, and a matrix equation of the type $Mc = b$ is formed, and subsequently solved with standard library methods as scipy. The matrix for a circle with $N \times N$ gridpoints is shown in equation 10.

$$\begin{pmatrix} m_{11} & \cdots & m_{1N^2} \\ \vdots & \ddots & \vdots \\ m_{N^21} & \cdots & m_{N^2N^2} \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_{N^2} \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ b_{N^2} \end{pmatrix} \tag{10}$$

The domain is a circular disk with radius 2, centered on the origin. On and beyond the boundary of this disk, the concentration is $c = 0$. At the point (0.6, 1.2) a source is present, so that the concentration there is 1.

III. Results & Conclusions

A. *Eigenmodes of membranes of different shapes*

In order to simulate the wave equation, we discretize the formula of equation 6.

$$v_{i,j}^{k+1} = \left(\frac{c\Delta t}{\Delta x}\right)^2 (v_{i+1,j}^k + v_{i-1,j}^k + v_{i,j+1}^k + v_{i,j-1}^k - 4v_{i,j}^k) + 2v_{i,j}^k - v_{i,j}^{k-1} \tag{11}$$

where k denotes the k-th iteration of the concentration ν, which iterates over the grid with i and j. c is the concentration, which takes a small timestep δt and space δx. $\delta x = L/N$ where L is the side length of the shape, and N the number of discretization steps. Moreover, to clarify the simulation, figure 1 is shown to explain the shape of a square drum and the corresponding matrix M of the eigenvalue problem (see table I). For example, we can see that node 1 and 4 are not connected to each other, which corresponds in matrix I. Furthermore, neither node 4 and 5, or node 8 and 9 are connected, since the nodes are on the other side of the squared membrane.

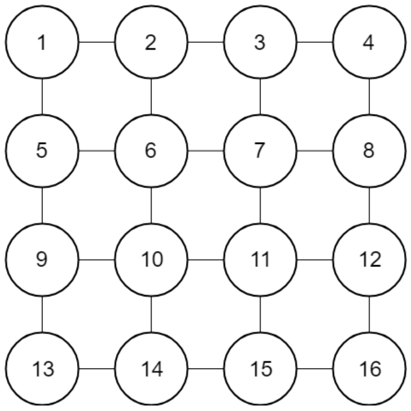


Fig. 1: The visual representation of a square drum with a 4 by 4 grid. The nodes show the points in the grid which include concentrations, and the edges illustrate the connections between the nodes. The boundary conditions are "reflective" or "no-flow", meaning that there is no diffusion over the edges of the system. We iterate from left to right.

We solved the eigenvalue problem using `scipy.linalg.eig()`. We used this function, because it gave good and reliable results and it did not take too long. When we started using more discretization steps (and thus creating larger matrices) we used `scipy.sparse.linalg.eigs()`. This method is much faster (see table II), but gives only a certain amount of eigenvalues, and that amount can not be more than $N - 1$, where N is the amount of rows or columns in the square matrix. Various plots of the eigenvectors ν with their frequencies are shown in figures 5-7. In these three figures, we did thus use the slower `scipy.linalg.eig()`, to be able to plot all the frequencies. The eigenmodes for a square (fig. 2) and circled shaped (fig. 4) drum sometimes show mirrored results, as the shapes are symmetric. The eigenmodes for a rectangular shaped drum are shown in figure 3. The eigenmodes of all shapes are comparable: more nodes and antinodes appear when we plot higher frequencies of the membrane of the drum.

The eigenfrequencies for each shape (square, rectangle, circle) of the drum as a function of L are very similar, and the graphs of the other shapes than the square can be found in our GitHub folder. The spectrum of eigenfrequencies slightly depends on the size L, see figure 5. When increasing the length of the drum, we see that we get more eigenfrequencies: this is due to the fact that the matrix is larger and we are thus solving for more eigenvalues. The interesting part is that we see lower frequencies, which holds true for real-life instruments: the bigger they get, the lower the frequency of the sound they make. By increasing the amount of discretization steps, we found higher and higher frequencies: the overtones of a drum. In other words, the frequencies we find depend on the number of discretization steps (when $L = 1$), illustrated in figure 6. In table III, we see the computational times for solving the eigenvalue problems with a different amount of discretization steps. As a result, the square and circular drum are calculated 8 times faster (for $N = 60$) than the rectangle drum, since the size of the matrix depends on the shape of the drum. When only looking at the first 20 eigenfrequencies (figure 7), we see that their distribution becomes more or less the same when the amount of discretization steps is bigger than 20. The same holds true for the rectangle and circled drum.

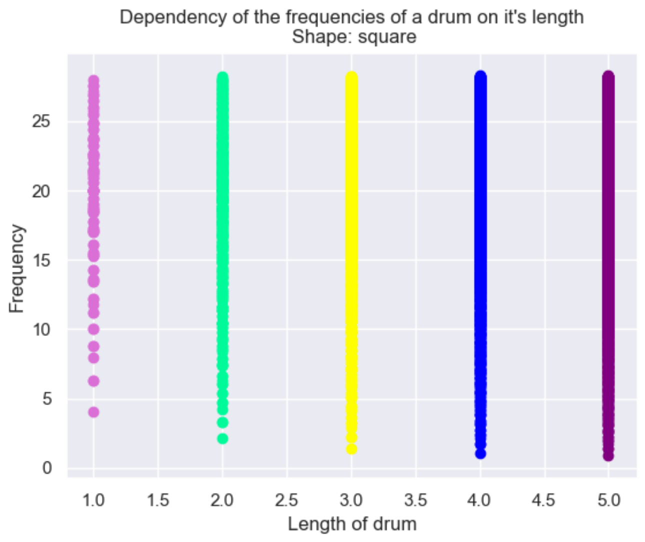


Fig. 5: Dependency of the eigenfrequencies of a square drum as a function of the length of the drum. The amount of discretization steps is $20 \times L$.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
1	**-4**	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0
2	**1**	**-4**	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0	0
3	0	**1**	**-4**	**1**	0	0	**1**	0	0	0	0	0	0	0	0	0
4	0	0	**1**	**-4**	0	0	0	**1**	0	0	0	0	0	0	0	0
5	**1**	0	0	0	**-4**	**1**	0	0	**1**	0	0	0	0	0	0	0
6	0	**1**	0	0	**1**	**-4**	**1**	0	0	**1**	0	0	0	0	0	0
7	0	0	**1**	0	0	**1**	**-4**	**1**	0	0	**1**	0	0	0	0	0
8	0	0	0	**1**	0	0	**1**	**-4**	0	0	0	**1**	0	0	0	0
9	0	0	0	0	**1**	0	0	0	**-4**	**1**	0	0	**1**	0	0	0
10	0	0	0	0	0	**1**	0	0	**1**	**-4**	**1**	0	0	**1**	0	0
11	0	0	0	0	0	0	**1**	0	0	**1**	**-4**	**1**	0	0	**1**	0
12	0	0	0	0	0	0	0	**1**	0	0	**1**	**-4**	0	0	0	**1**
13	0	0	0	0	0	0	0	0	**1**	0	0	0	**-4**	**1**	0	0
14	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	**-4**	**1**	0
15	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	**-4**	**1**
16	0	0	0	0	0	0	0	0	0	0	0	**1**	0	0	**1**	**-4**

TABLE I: The connection matrix M of a square drum encoding the connections of a 4 by 4 grid. The boundary conditions are “reflective” or “no-flow”, meaning that there is no diffusion over the edges of the system: everything beyond the membrane of the drum is considered to be 0 and not moving. 0 denotes no connection, 1 a connection, and the -4 is referring to its own node. The colored boxes show the repeating parts of this matrix, which is used for implementation of the program.

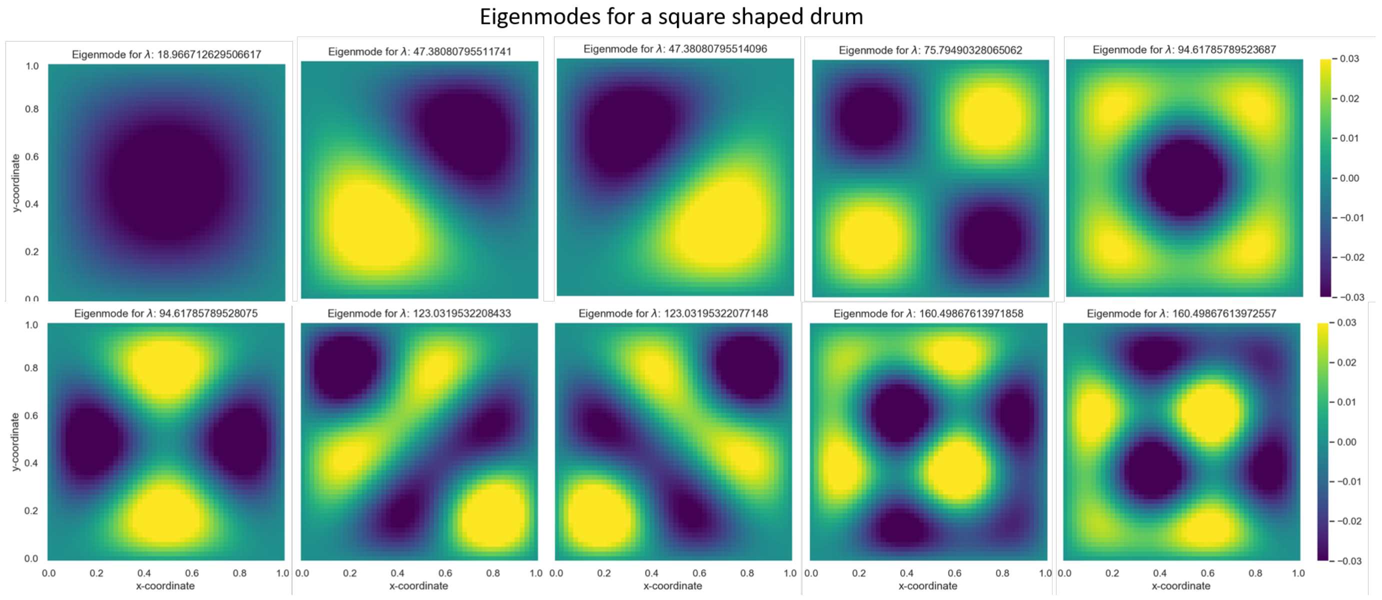


Fig. 2: The first ten eigenmodes of a square drum with side length $L = 1$. Eigenfrequencies belonging with these λ 's are $4.36, 6.88, 6.88, 8.71, 9.73, 9.73, 11.09, 11.09, 12.67, 12.67$, respectively. The amount of discretization steps is $N = 50$, $\delta x = 0.02$.

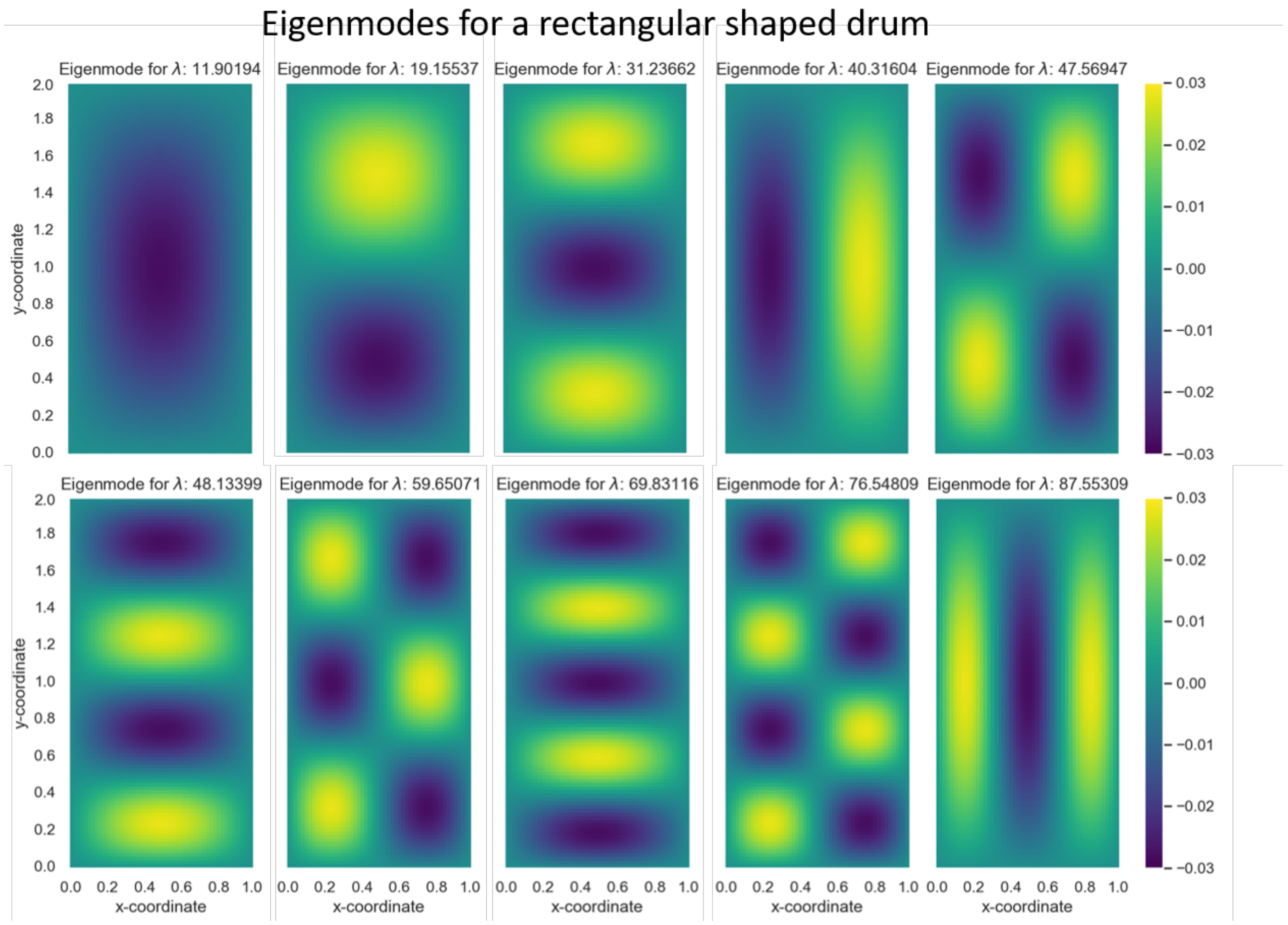


Fig. 3: The first ten eigenmodes of a rectangular drum with sides L and $2L$, and length $L = 1$. Eigenfrequencies belonging with these λ's are $3.45, 4.38, 5.59, 6.35, 6.90, 6.94, 7.72, 8.36, 8.75, 9.36$, respectively. The amount of discretization steps is $N = 50 \times L$, $\delta x = 0.02$ (this means that there are 100 steps in the y-direction).

	Computation time (s)		
Method	10 runs, N = 10	10 runs, N=25	10 runs, N = 50
scipy.linalg.eig(M)	0.49	7.25	222.87
scipy.sparse.linalg.eigs(M, k=10, which='LR')	0.25	2.29	49.68

TABLE II: Computation times of solving the eigenvalue problem, with different N. We see that the difference between the two methods gets bigger when N increases. The sparse solver can make the computation time significantly shorter.

	Amount of discretization steps											
	5	10	15	20	25	30	35	40	45	50	55	60
Solve time for rectangle drum (s)	0.00	0.08	0.36	1.31	3.56	9.10	21.01	41.15	75.20	149.03	260.62	415.50
Solve time for square drum (s)	0.01	0.08	0.12	0.30	0.90	1.99	4.49	7.91	12.82	21.72	40.18	57.19

TABLE III: Computational times for solving the eigenvalue problem of matrices that belong with the rectangle or square drum. The circular solving times are very similar to that of a square drum, because the matrix has the same size.

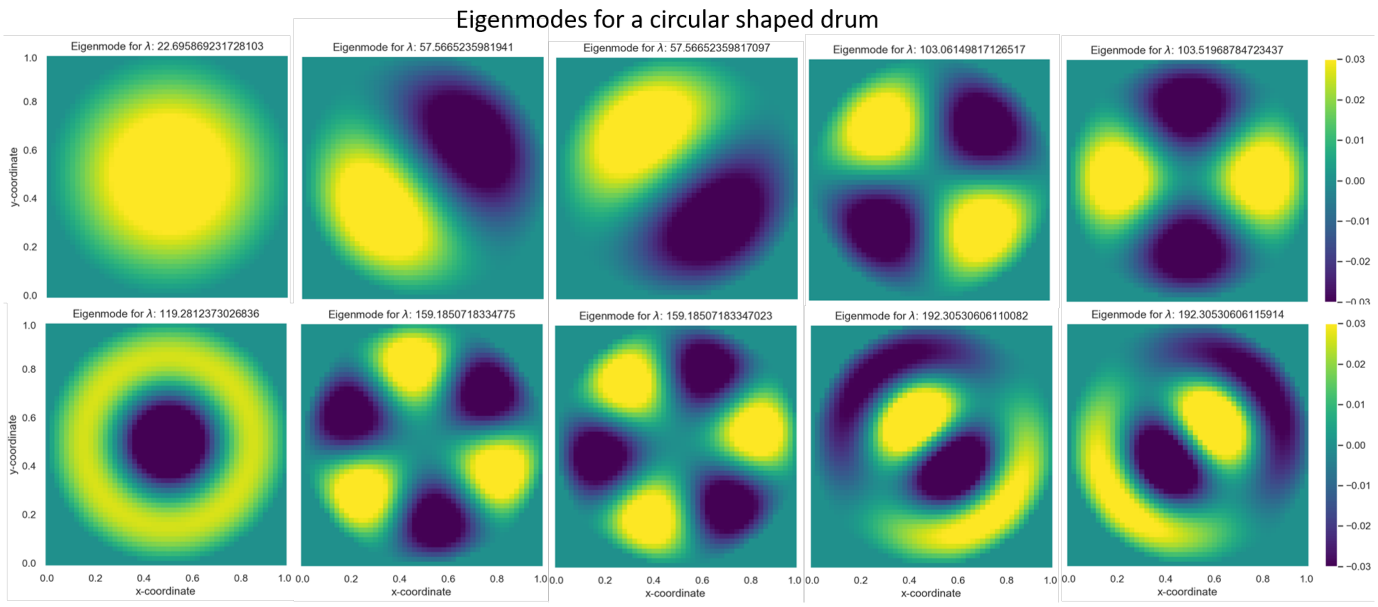


Fig. 4: The first ten eigenmodes of a circular drum with diameter L, and length $L = 1$. Eigenfrequencies belonging with these λ's are $4.76, 7.59, 7.59, 10.15, 10.15, 10.92, 12.62, 12.62, 13.87, 13.87$, respectively. The amount of discretization steps is $N = 50$, $\delta x = 0.02$.

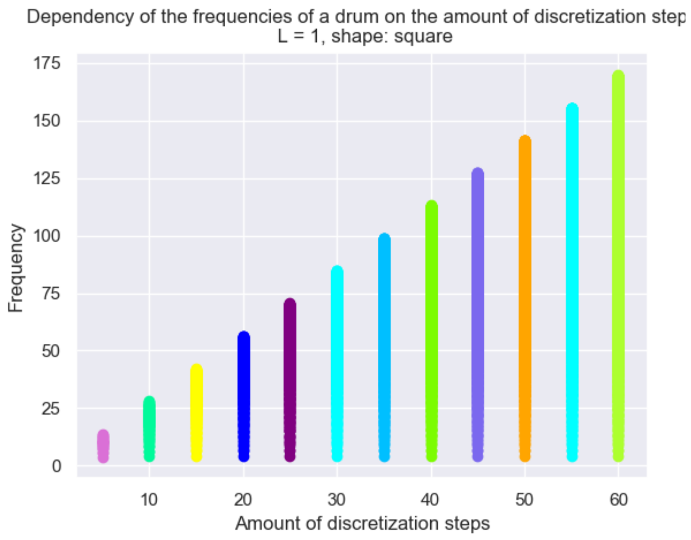


Fig. 6: Dependency of all eigenfrequencies of a square drum as a function of the amount of discretization steps. All eigenfrequencies are plotted. The length of the drum is $L = 1$.

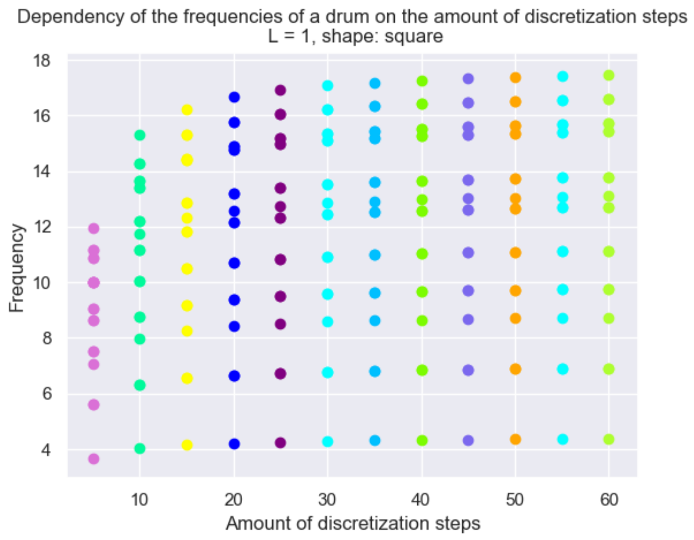
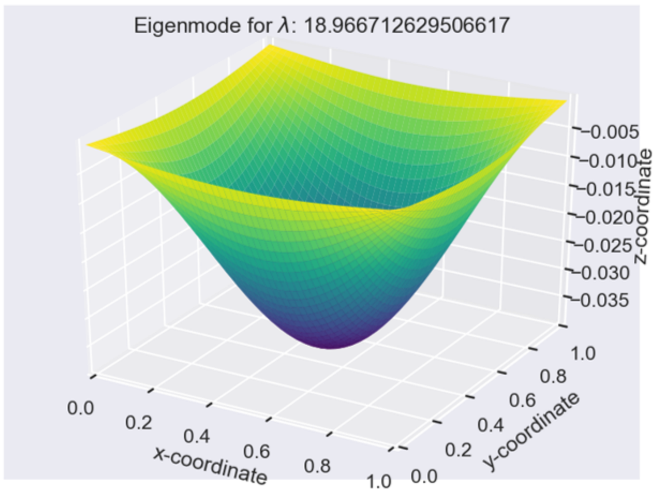


Fig. 7: Dependency of all eigenfrequencies of a square drum as a function of the amount of discretization steps. The first 20 eigenfrequencies are plotted. The length of the drum is $L = 1$.

The behavior of a couple of the first eigenmodes of the differently shaped drums are shown with an animated plot through time. See the set3/results/movies section of our GitHub page for the mp4's. In figure 8, 3D plots of the first and fifth eigenmode of a square drum are shown. Besides that, in figure 9 a 3D plot of the circular drum is also shown, to show the difference in the shape of these drums. These solutions show how the dynamics of the eigenmode of the drum changes in time. Looking at the square ($\lambda = 18.97$) drum, the plot shows the corresponding shape of the figure, as this is the first eigenmode. The circled ($\lambda = 119.28$) drum plot also shows the corresponding shape of the figure. Considering the animations, the eigenmode oscillates in time, but returns to the same shape of the eigenmode, which is especially clearly shown for the square drum ($\lambda = 94.62$).



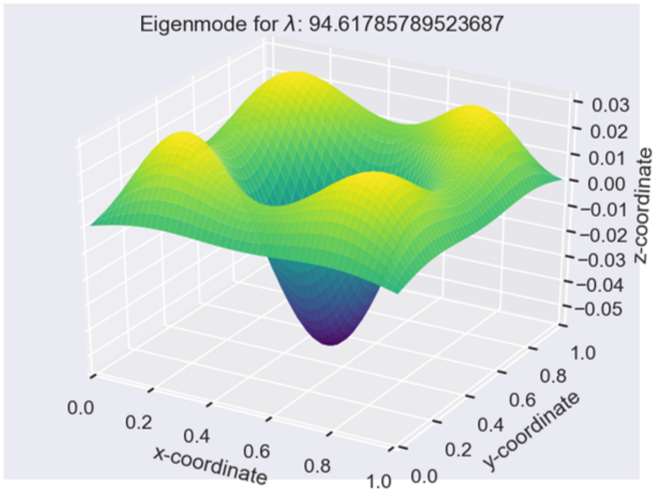


Fig. 8: Two eigenmodes of a square drum are shown in 3D. Eigenvalues $\lambda = 18.97$ and $\lambda = 94.62$, with frequencies 4.36 and 9.73 respectively, are shown. $c = 1$, $\delta x = 0.02$.

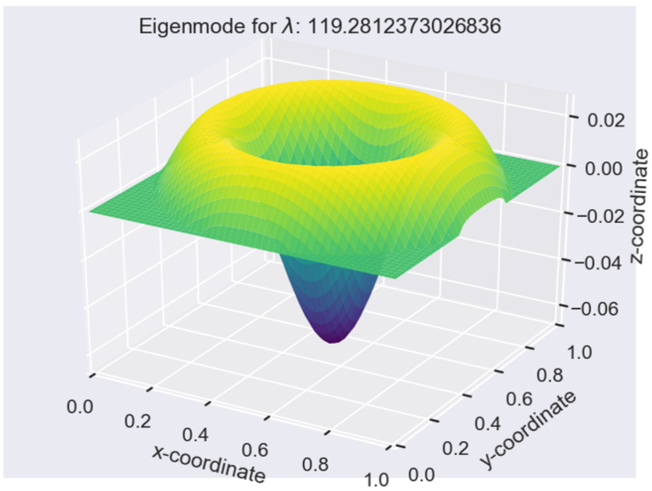


Fig. 9: Eigenmode of a circular drum shown in 3D. Eigenvalue $\lambda = 119.28$, with frequency 10.92. $c = 1$, $\delta x = 0.02$.

B. *Direct methods for solving steady state problems*

The steady state concentration $c(x, y)$ is calculated by discretizing the diffusion equation (see equation 9) as mentioned in the methods section. The discrete diffusion equation becomes:

$$c_{i,j}^{k+1} = c_{i,j}^{k} + \frac{\delta t D}{\delta x^2} (c_{i+1,j}^{k} + c_{i-1,j}^{k} + c_{i,j+1}^{k} + c_{i,j-1}^{k} - 4c_{i,j}^{k}) \quad (12)$$

where k denotes the k-th iteration of the the concentration c, which iterates over the grid with i and j. The concentration is calculated as a function of the coordinates x and y and time t. D is the diffusion constant. This explicit scheme is stable if:

$$\frac{4\delta t D}{\delta x^2} \leq 1 \quad (13)$$

This determines the maximum time step δt that we can take. The matrix equation $Mc = b$ is formed for a circular disk with radius 2, and $\delta x = 0.1$. This equation is solved with standard methods from the scipy library, and no iterative or sparse solver was needed for this system. The solution is plotted in figure 10.

First, we construct matrix M and the initial vector b in order to subsequently solve the matrix equation and derive vector c. Matrix M is constructed by first

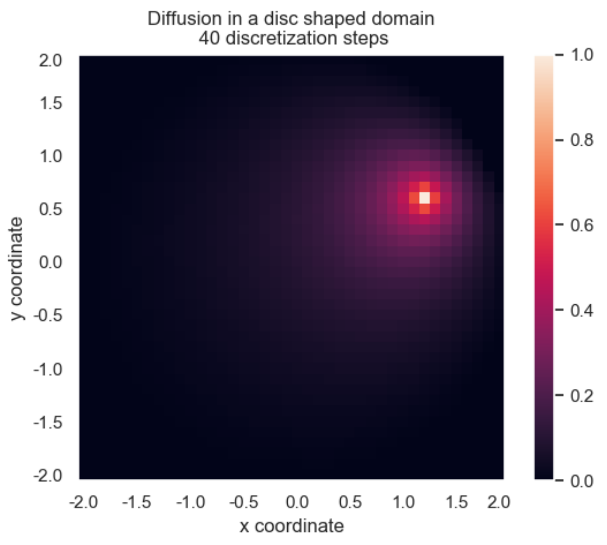


Fig. 10: The steady state of diffusion on a circular disk with radius 2. The source is placed at (0.6, 1.2), and the concentration of this source is $c = 1$ and is kept constant. Around the circular disk, sinks are present with a concentration of $c = 0$. $\delta x = 0.1$, $\delta t = 0.001$.

implementing all the diagonals, which are multiplied with a constant, the same way we did for a square (see table I. Based on the shape of a circle, we check whether a row in the matrix belongs to a point in the circle, and update the diagonal of the matrix M when either in the circle or the source: these points do not diffuse. The mentioned rows are updated to contain all 0s and only a single 1 on the diagonal. Vector b is basically the initial condition of the concentration matrix, which we flatten, because we need a vector. After creating M and b, we solve the matrix equation and reshape vector c to the size of the grid.

IV. Discussion

We discretized the wave equation in two dimensions and showed a visual representation of a square drum with a 4 by 4 grid and the corresponding connection matrix. This could also be done for a membrane with a rectangle or circle.

Using the method `scipy.linalg.eig()` and `scipy.sparse.linalg.eigs()`, we successfully solved the eigenvalue problem. The sparse solver can make the computation time significantly shorter for solving the eigenvalue problem. Maybe we could have tried `scipy.sparse.linalg.eigsh()` to make it even faster, since we thought our matrix was compatible with this kind of library method.

Lower frequencies appear in the eigenfrequencies spectrum when we increase L. How many frequencies (of overtones) we find depends on the number of discretization steps. The number of discretization steps indeed determines the spectrum of eigenfrequencies, since this changes the size of the connection matrix and hence the spectrum of eigenfrequencies. This holds for all types of shapes of the drum we encountered.

Moreover, we animated the time-dependent solutions of the wave equation for the circle and square shaped drum with different eigenmodes. The animation clearly shows how the eigenmodes behave in time.

By discretizing the diffusion equation, the steady state concentration $c(x, y)$ was plotted. The concentration did not change outside the circle, as boundary conditions were applied in the matrix equation. The computation time of the solution could possibly be improved by using a different direct method, i.e. by solving the matrix in a more efficient way with for instance (parallel) software libraries.