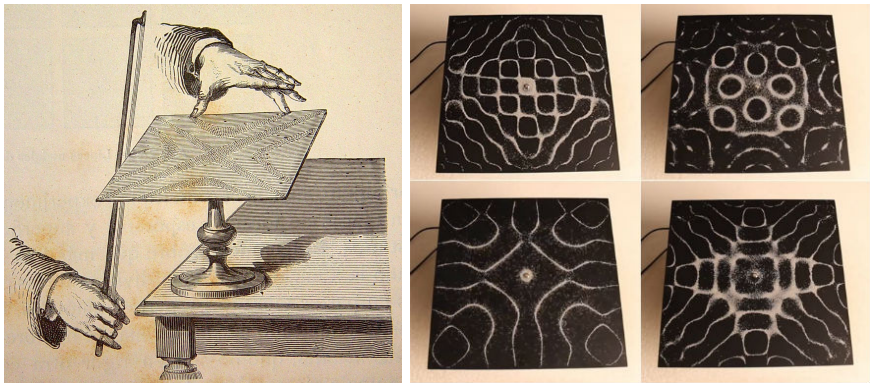
SciComp Project 2
Week 3 (5 points)

September 19, 2022
To be handed in Monday September 26 before 12:00 noon

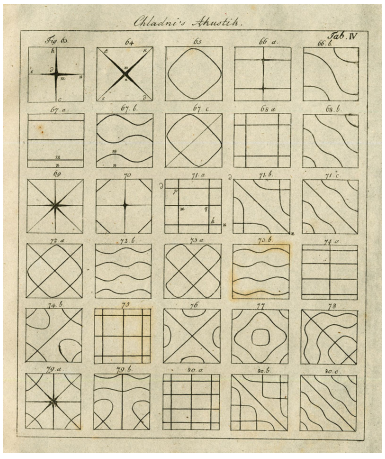


1 Background

Among the great pioneers of computational physics was Walther Ritz, who (half a century before the digital computer) invented a computational method for solving tough differential equations that is among the most used today: the *Ritz-Galerkin*-method. Whether solving quantum systems in Hilbert space or simulating whether a bridge will crash in a storm, his method usually lies underneath.

We will not look at that until later in the course. Instead, we will study the problem he invented it for: The mysterious harmonics of Chladni plates, one of the first and most visually striking experiments that let us *see* sound.

Ernst Chladni, a musician and physicist from 18^{th} century Wittenberg, discovered that driving a metal plate at different frequencies with his violin bow sometimes caused dust to gather into beautiful patterns. At most frequencies, the dust was just shaken about, but through a systematic approach with sand sprinkled over the plate, he found that the magic happened at certain frequencies, which he carefully drew and collected.



Chladni's drawing of resonance modes he discovered with iron plate, violin bow, and sand.

Already at Chladni's time, it was well-understood why the figures appeared: The special frequencies were *resonances*, harmonics inherent to the plate geometry that lead to standing waves. In the *nodes* (the regions where the wave is zero) the sand settles; everywhere else it is pushed around until finding a nodal region to rest. [1] But it took over a hundred years before anyone could solve the equations and predict the patterns.

[1] The same principle is used today with 3-dimensional sound waves in air for *acoustic levitation*.

The thin plates are ruled by Kirchhoff's dynamics (1850), time-evolved by the *biharmonic operator*

$$\Delta^2 = \left(\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} + \frac{\partial^4}{\partial x^2 \partial y^2} \right) \tag{1}$$

similar, but not the same as the wave equation. The resonance modes are the eigenfunctions of the biharmonic operator, and the resonance frequencies correspond to the eigenvalues:

$$\Delta^2 u(x,y) = \lambda u(x,y) \tag{2}$$

under free boundary conditions, subject only to the plate being a solid, elastic object:

$$\begin{array}{lll} x\text{-boundary:} & \frac{\partial}{\partial x}\left(\frac{\partial^2}{\partial x^2} + (2-\mu)\frac{\partial^2}{\partial y^2}\right) u(x,y) = 0 & \text{and} \quad \left(\frac{\partial^2}{\partial x^2} + \mu\frac{\partial^2}{\partial y^2}\right) u(x,y) = 0 \\ y\text{-boundary:} & \frac{\partial}{\partial y}\left(\frac{\partial^2}{\partial y^2} + (2-\mu)\frac{\partial^2}{\partial x^2}\right) u(x,y) = 0 & \text{and} \quad \left(\frac{\partial^2}{\partial y^2} + \mu\frac{\partial^2}{\partial x^2}\right) u(x,y) = 0 \end{array} \tag{3}$$

where μ is an elasticity constant, a material property of the plate.

2 Data

The full calculation is beyond the scope of this week:[2] **You will only need to calculate the resonance eigenvectors and eigenvalues** from a matrix representation $\mathbf{K}$ of the biharmonic operator, which you may download here: Chladni-Kmat.npy. That is, you will simply be working with the eigenproblem:

$$\mathbf{K}\mathbf{x} = \lambda \mathbf{x} \tag{4}$$

The matrix representation $\mathbf{K}$ incorporates also the boundary conditions, so that eigenvectors will be solutions to the full problem.

In Python you load the matrix as `Kmat = np.load("Chladni-Kmat.npy")`.

For testing your implementation, you can use the matrices given in examplematrices.py. Simply report the results for the highest-numbered matrix that works.

Visualization

To show your solutions, download chladni_show.py and the data file chladni_basis.npy for Python[3]. Running `chladni_show` defines the `basis_set`, a $15 \times 500 \times 500$ array of 15 basis functions each defined on a 500×500 grid, and the following three functions:

- `show_waves(x,basis_set)`: Shows the actual wavefunction corresponding to coefficient vector `x`
- `show_nodes(x,basis_set)`: Shows the wavefunction zeros, where the sand gathers.
- `show_all_wavefunction_nodes(T,lambdas,basis_set)`: Shows the zeros of all the eigenfunctions defined by the columns of `T`.

[2] If you are interested in how it is done, Gander and Wanner's From Euler, Ritz, and Galerkin to Modern Computing has an excellent treatment of the problem. My matrix-representation of the problem, $\mathbf{K}$, is calculated by a similar method, with a slightly different and larger basis to yield more modes.

[3] Load with `np.load()`

3 Questions for Week 3

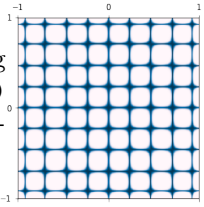
a. (1) Implement a function `centers,radii = gershgorin(A)` that locates the eigenvalues of a matrix using Gershgorin's theorem. (2) Localize the eigenvalues of $\mathbf{K}$, and report the disk centers and radii.

b. (1) Implement a function `lambda = rayleigh_qt(A,x)`, which takes a matrix $\mathbf{A}$ and approximate eigenvector $\mathbf{x}$, and returns the approximate eigenvalue λ given by the Rayleigh quotient.

(2) Implement a function `x, k = power_iterate(A,x0)` for power iteration, which takes a matrix $\mathbf{A}$ and an initial vector $\mathbf{x}_0$ as arguments, and returns an eigenvector $\mathbf{x}$ together with the number k of iterations used. Don't forget to choose and implement a suitable convergence criterion.

(3) Test it by finding the largest eigenvalue of the example matrices. Report the eigenvalue found, the Rayleigh residual (the residual of the Raleigh quotient viewed as a least-squares system), and the number k of iterations used.

(4) What is the largest eigenvalue of $\mathbf{K}$? Visualize your eigenfunction using `show_waves(x,basis_set)` to see the wave, and `show_nodes(x,basis_set)` to see where the sand will gather. The eigenfunction for the largest eigenvalue should have nodes along an 8×8 grid.



c. (1) Write a Rayleigh quotient iteration function `x, k = rayleigh_iterate(A,x0, shift0)`, which takes a matrix $\mathbf{A}$, an initial vector $\mathbf{x}_0$, and an approximate eigenvalue $\texttt{shift}_0$ as arguments and returns an eigenvector $\mathbf{x}$ together with the number k of iterations used.

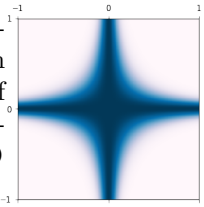
The multiplication by the inverse matrix should be implemented using either your own LU-factorization from as `x = lu_solve(A,y)` or QR-factorization from as `x = qr_solve(A,y)`.

NB: If you did not get either to work in previous weeks, you can use an in-built linear solver and make a small note that you do this.

(2) Test it with the example matrices, and report the eigenvalues found, Rayleigh residual, and the number k of iterations used.

d. An important feature of inverse iteration and Rayleigh-iteration is the ability to calculate *any* eigenvalue and its eigenvectors: given an approximate starting point, we obtain an eigenvector to the nearest eigenvalue.

(1) Why can you not get all the eigenvalues and -vectors with pure power-iteration? (2) Use your Rayleigh quotient iteration function together with the Gershgorin centers to calculate as many eigenvectors and eigenvalues of $\mathbf{K}$ as you can. Are you able to get all of them? Why? If not, find the remaining one(s) by any means necessary. Check using `show_nodes(x,basis_set)` that the eigenfunction with lowest eigenvalue looks like a cross:



(3) Construct the transformation matrix $\mathbf{T}$ whose columns are the eigenvectors in order of ascending eigenvalues, and check that $\mathbf{K} = \mathbf{T}\Lambda\mathbf{T}^{-1}$ with diagonal Λ.[4] (4) Visualize your solutions using the provided function `show_all_wavefunction_nodes(T,lambdas,basis_set)`.

[4] You may use the system library inverse for this.