Project 2

Lars Opgård & Sunniva Kiste Bergan

(Dated: September 27, 2022)

https://github.com/sunnikbe/comfys/tree/main/project2

THE BUCKLING BEAM

We want to see how a beam of some arbitrary material will buckle under pressure from a force. To do this we first make some assumptions and simplifications. All information given below was retrieved from the course page FYS3150/FYS4150 course material[1].

The beam has length L described by the x-axis as $x \in [0, L]$. The function $u(x)$ will then represent the vertical displacement. We assume that the endpoints of the beam are pinned, i.e. they can rotate freely, but they stay fixed at $y = 0$. We also assume that the force is applied parallel to the beam in the negative x direction.

The second-order differential equation,

$$\gamma \frac{d^2u(x)}{dx^2} = -Fu(x), \tag{1}$$

where γ is a constant given by material properties, describes the buckling beam´s possible configurations, but we want to make further simplifications.

We want to make the solutions dimensionless and solve the problem numerically.

Our equation then becomes

$$\frac{d^2u(\hat{x})}{d\hat{x}^2} = -\lambda u(\hat{x}), \tag{2}$$

where $\hat{x}$ is our dimensionless variable defined by $\hat{x} \equiv \frac{x}{L}$ and $\lambda = \frac{FL^2}{\gamma}$. $\hat{x} \in [0, 1]$.

By discretizing eq. 2, we will have an approximation to $u(x)$ denoted by $v(x)$. Our discretized problem will be described as

$$\frac{-v_{i+1} + 2v_i - v_{i-1}}{h^2} = \lambda v_i,$$

for $i = 0, 1, ..., n$, where h is our step size. We want to solve this equation as an eigenvalue problem

$$\mathbf{A}\vec{v} = \lambda\vec{v}, \tag{3}$$

where $\mathbf{A}$ is a tridiagonal matrix (a, d, a) with $a = -1/h^2$ and $d = 2/h^2$. By solving eq. 3 using the Jacobi rotation algorithm we end up with multiple solutions made up by pairs of eigenvalues λ and eigenvectors $\vec{v}$.

When visualizing our results we want to pick out the three eigenvectors corresponding to the three lowest eigenvalues of our numerical solution (Jacobi rotation) and plot them against $\hat{x}$. This will provide a plot showing three different configurations our beam could have.

PROBLEM 1

From eq. 1 we can show that using the dimensionless units $\hat{x} = x/L$ we get the new eq. 2. If we input $\hat{x}$ into eq. 1 and use the chain-rule we get

$$\gamma \frac{d^2u(\hat{x})}{dx^2} = \gamma \frac{d^2u(\hat{x})}{d\hat{x}^2}\frac{d^2\hat{x}^2}{dx^2} = \frac{\gamma}{L^2}\frac{d^2u(\hat{x})}{d\hat{x}^2}.$$

We then rewrite it as

$$\frac{d^2u(\hat{x})}{d\hat{x}^2} = -\frac{FL^2}{\gamma}u(\hat{x}) = -\lambda u(\hat{x}),$$

where $\lambda = \frac{FL^2}{\gamma}$.

PROBLEM 2

A program (tridiag.cpp) that sets up the tridiagonal matrix $\mathbf{A}$ for $N = 6$, solves eq. 3 and checks the results against the analytical results, is found at our github repo. The analytical eigenvalues for a NxN tridiagonal matrix are calculated as such

$$\lambda^{(i)} = d + 2a\cos\left(\frac{i\pi}{N+1}\right), \; i = 1, ..., N.$$

And the eigenvectors

$$\vec{v}^{(i)} = \left[\sin\left(\frac{i\pi}{N+1}\right), ..., \sin\left(\frac{Ni\pi}{N+1}\right)\right]^T, \; i = 1, ..., N.$$

Running the program we get the eigenvalues for a 6x6 symmetric tridiagonal matrix with all the elements along the diagonal equal to 2 and all the elements along the sub- and superdiagonal equal to -1. A comparison is found in table I.

Analytical	Armadillo
0.1981	0.1981
0.7530	0.7532
1.5550	1.5550
2.4450	2.4450
3.2470	3.2470
3.8019	3.8019

TABLE I. Table comparing eigenvalues for a 6x6 tridiagonal matrix solved using the analytical and the numerical (Armadillo) versions of the Jacobi rotation method.

Below, the eigenvectors from the analytical result is listed

$$\begin{bmatrix} 0.2319 & 0.4179 & 0.5211 & 0.5211 & 0.4179 & 0.2319 \\ 0.4179 & 0.5211 & 0.2319 & -0.2319 & -0.5211 & -0.4179 \\ 0.5211 & 0.2319 & -0.4179 & -0.4179 & 0.2319 & 0.5211 \\ 0.5211 & -0.2319 & -0.4179 & 0.4179 & 0.2319 & -0.5211 \\ 0.4179 & -0.5211 & 0.2319 & 0.2319 & -0.5211 & 0.4179 \\ 0.2319 & -0.4179 & 0.5211 & -0.5211 & 0.4179 & -0.2319 \end{bmatrix}$$

The corresponding eigenvectors from Armadillo are

$$\begin{bmatrix} 0.2319 & -0.4179 & 0.5211 & 0.5211 & 0.4179 & -0.2319 \\ 0.4179 & -0.5211 & 0.2319 & -0.2319 & -0.5211 & 0.4179 \\ 0.5211 & -0.2319 & -0.4179 & -0.4179 & 0.2319 & -0.5211 \\ 0.5211 & 0.2319 & -0.4179 & 0.4179 & 0.2319 & 0.5211 \\ 0.4179 & 0.5211 & 0.2319 & 0.2319 & -0.5211 & -0.4179 \\ 0.2319 & 0.4179 & 0.5211 & -0.5211 & 0.4179 & 0.2319 \end{bmatrix}$$

PROBLEM 3

(a) In our main.cpp code a function called "largest off diagonal element" (line 91) returns the largest off-diagonal element of the input matrix.

(b) A test code for the function "largest off diagonal element" is found in line 60 of our main.cpp code.

Testing this code for a simple 4x4 symmetric matrix **A**

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & -0.7 & 0 \\ 0 & -0.7 & 1 & 0 \\ 0.5 & 0 & 0 & 1 \end{bmatrix}$$

Running the code for this matrix yields the absolute largest off diagonal value in the matrix:
max value = 0.7.
We can then also read from the upper off diagonal that the matrix element is $A(2, 3)$, which we will set to $k = 2$ and $l = 3$ which we will use when solving the Jacobi rotation algorithm.

PROBLEM 4

(a) The Jacobi rotation algorithm for solving eq. 3 is declared in line 115 and 161 of main.cpp.

(b) A test of the Jacobi rotation algorithm for $N = 6$ is found at line 76 in main.cpp.

Running the test with the same matrix from problem 2 we get the eigenvalues:

$$\begin{matrix} 0.1981 \\ 0.7530 \\ 1.5550 \\ 2.4450 \\ 3.2470 \\ 3.8019 \end{matrix}$$

and the eigenvectors as the columns in the matrix:

$$\begin{bmatrix} 0.2319 & -0.4179 & -0.5211 & 0.5211 & 0.4179 & -0.2319 \\ 0.4179 & -0.5211 & -0.2319 & -0.2319 & -0.5211 & 0.4179 \\ 0.5211 & -0.2319 & 0.4179 & -0.4179 & 0.2319 & -0.5211 \\ 0.5211 & 0.2319 & 0.4179 & 0.4179 & 0.2319 & 0.5211 \\ 0.4179 & 0.5211 & -0.2319 & 0.2319 & -0.5211 & -0.4179 \\ 0.2319 & 0.4179 & -0.5211 & -0.5211 & 0.4179 & 0.2319 \end{bmatrix}$$

PROBLEM 5

(a) Our program was run with different choices of N, and our scaling data is presented in blue in figure 1. The relation between the number of iterations and N is described by a polynomial function of the second degree.

(b) The scaling behavior of **A** as a dense matrix is shown in orange (dotted line) in figure 1. It was to be expected that the tridiagonal matrix (blue) would have less iterations. The discrepancy between the number of iterations is more evident for higher values of N for the two different matrices.

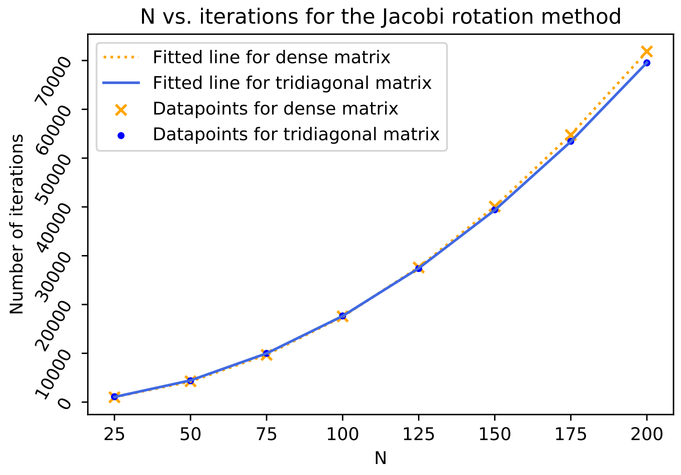

FIG. 1. A plot showing how the number of iterations of the Jacobi rotation method is related to the size of the matrix, which is NxN. The blue dots represents the datapoints of the method used on a tridiagonal matrix and the orange crosses represents the same for a dense matrix. Both have been fitted with a line to show how the number of iterations vary with N.

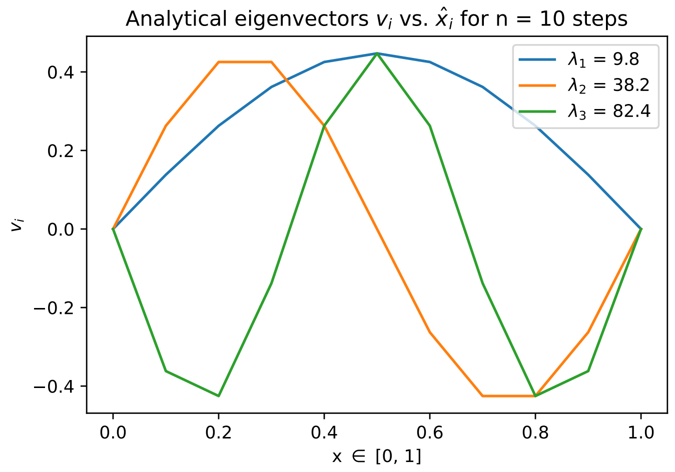

FIG. 2. Plot showing three different solutions (eigenvectors) to our Jacobi code corresponding to the three lowest eigenvalues λ_1, λ_2 and λ_3 for $n = 10$ steps.

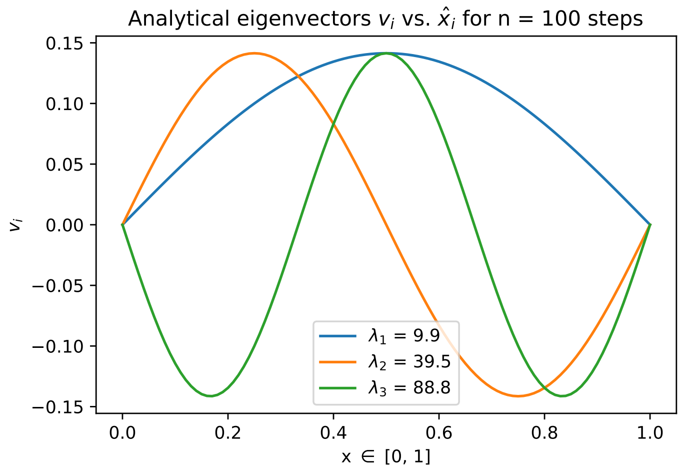

FIG. 3. Plot showing three different solutions (eigenvectors) to our Jacobi code corresponding to the three lowest eigenvalues λ_1, λ_2 and λ_3 for $n = 100$ steps.

PROBLEM 6

(a) A plot of our Jacobi code for $n = 10$ steps is shown in figure 2. The plot shows three different solutions corresponding to the three lowest eigenvalues; λ_1, λ_2 and λ_3. The eigenvector elements v_i of the eigenvectors of λ_1, λ_2 and λ_3 respectively are plotted along the y - axis. The discretized x - values are plotted along the x - axis.

(b) The same plot for discretization of $\hat{x}$ with $n = 100$ steps is shown in figure 3.

REFERENCES

[1] FYS3150/FYS4150 course material - Project 2. https://anderkve.github.io/FYS3150/book/projects/project2.html# (accessed: 15.09.2022)