Project 2

Laila Egbeocha Andersland
(Dated: 27. september 2021)

https://github.com/laila503/fys3150

PROBLEM 1

We have the second-order differential equation

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

and we have the unitless variable $\hat{x} = x/L$ So that $x = L\hat{x}$. Well insert this eq.(1):

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

Since L is the length of the beam it is a constant, we can extract it from the function:

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

Rearranged:

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

And set $\lambda = \frac{FL^2}{\gamma}$, so we'll get:

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

PROBLEM 2

We assume that $\vec{v}_i$ is a set of orthonormal basis vector, and $\mathbf{U}$ is an orthogonal transformation matrix. To show that transformations with $\mathbf{U}$ preserves orthonormality, we will show that $\vec{w}_i = \mathbf{U}\vec{v}_i$ i is also an orthonormal set.

In addition to $\vec{w}_i$ we introduce $\vec{w}_j$ and apply the inner product between them:

$$\vec{w}_i \cdot \vec{w}_j = (\mathbf{U}\vec{v}_i)^T(\mathbf{U}\vec{v}_j)$$

And since the the transpose rule gives $(\mathbf{U}\vec{v}_i)^T = \vec{v}_i{}^T\mathbf{U}^T$ We can apply this to the inner product:

$$\vec{w}_i \cdot \vec{w}_j = (\mathbf{U}\vec{v}_i)^T(\mathbf{U}\vec{v}_j) = \vec{v}_i^T\mathbf{U}^T\mathbf{U}\vec{v}_j$$

And since $\mathbf{U}^T\mathbf{U} = \mathbf{U^{-1}U} = \mathbf{I}$ we'll have:

$$\vec{v}_i^T\mathbf{U}^T\mathbf{U}\vec{v}_j = \vec{v}_i^T\mathbf{I}\vec{v}_j = \vec{v}_i \cdot \vec{v}_j = \vec{v}_i^T\vec{v}_j = \delta_{ij}$$

This shows that $\vec{w}_i$ is orthonormal.

PROBLEM 3

To set up a tridiagonal matrix we use the *Helper functions for creating tridiagonal matrices* in the *Code Snippets* -section [1]. We use Armadillo to find the eigenvectors and eigenvalues with the function `find_analytical_eigvec` and `find_analytical_eigval`. Lastly, we check and verify that the numbers agree for the given matrix A.

PROBLEM 4

We create a function that looks through all off-diagonal elements, and saves the largest element and its indices. This function can be found as `max_offdiag_symmetric` in `eigensolver.cpp`. We then made a test using a known matrix, which raises an error if the function gives the wrong answer. This function is called `test_code_max`.

PROBLEM 5

Here we follow the approach from the lecture book in chapter 7.4 Jacobi's method [2] and use the code snippets as a guide for how to structure the code. We make the function `jacobi_rotate` to perform a single Jacobi rotation, and then the function `jacobi_eigensolver` will run the rotations as long as the maximum off-diagonal value is less than an set ϵ and the number of iterations is less than a set maximum value.

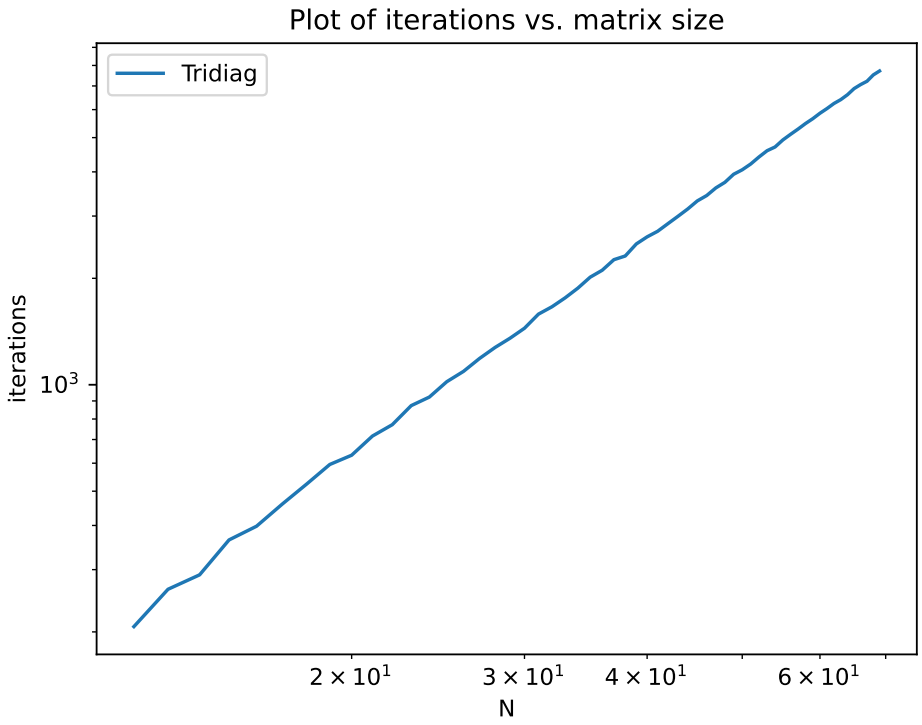
To check if this gives the correct answer, we have here written the eigenvalues for $N = 6$. As we can see, to at least the third decimal, they are identical.

```
Jacobi      Analytical
7.130       7.130
27.109      27.109
55.978      55.978
88.022      88.022
116.891     116.891
136.870     136.870
```

PROBLEM 6

a

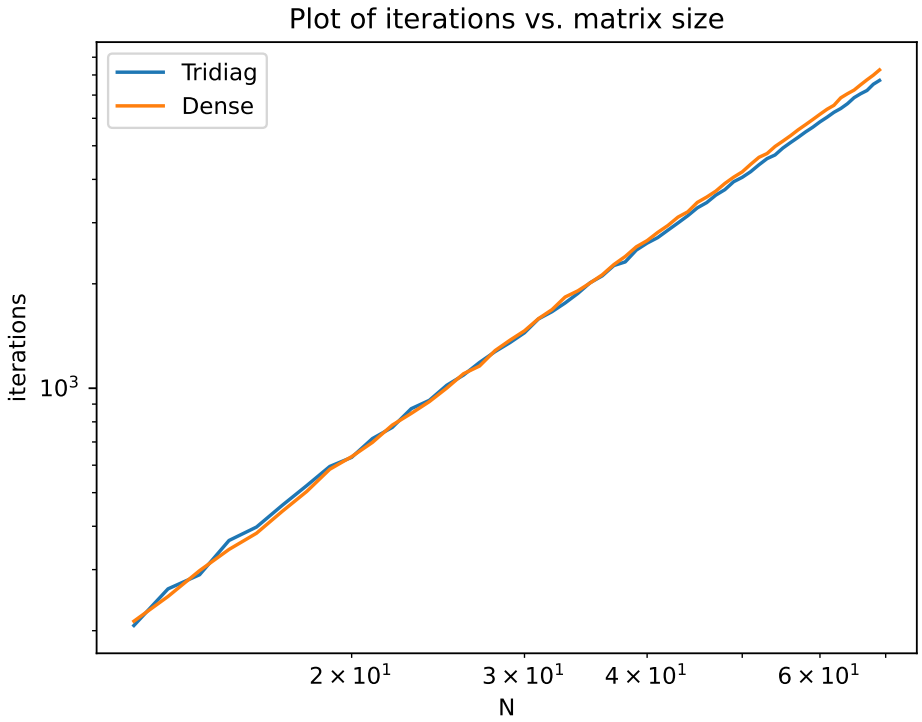
We now try to estimate the number of iterations needed to find the eigenvalues/vectors, compared to the matrix size N, and the result is shown in figure(1).



Figur 1. A loglog plot of number of iterations vs. the size of the tridiagonal matrix. The iterations refers to the number of times the Jacobi rotations are done in the Jacobi method for finding the eigenvalues and eigenvectors. We see that the number of iterations grows approximately as N^2, which is to be expected.

b

At first glance, it would be expected that the number of similarity transformations would be larger for a dense matrix. However when we plot the scaling behavior for one sparse and one dense matrix of the same size, see figure(2), we see that it is more or less the same. The reason for this is that when one similarity transformation is done, for the sparse, tridiagonal matrix, it will spread the numbers from the diagonal and out on the other places, making it more and more "dense". So, it will not matter what the initial matrix look like anymore.



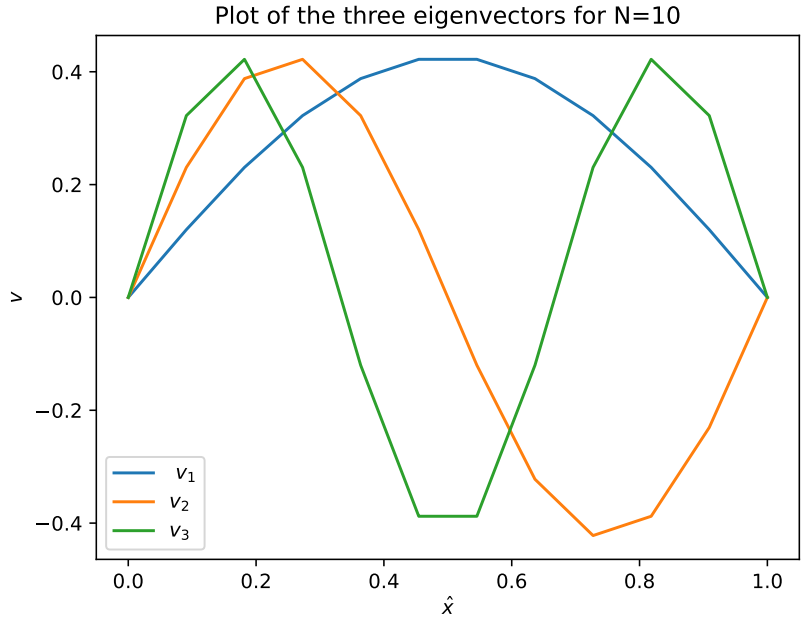
Figur 2. A loglog plot of number of iterations vs. the size of the tridiagonal matrix and a dense matrix. We expected to see lower number of iterations needed for the tridiagonal matrix, since it is more sparse and have less numbers, but it seems that it is required almost the same amount of iterations as the dense matrix.

7

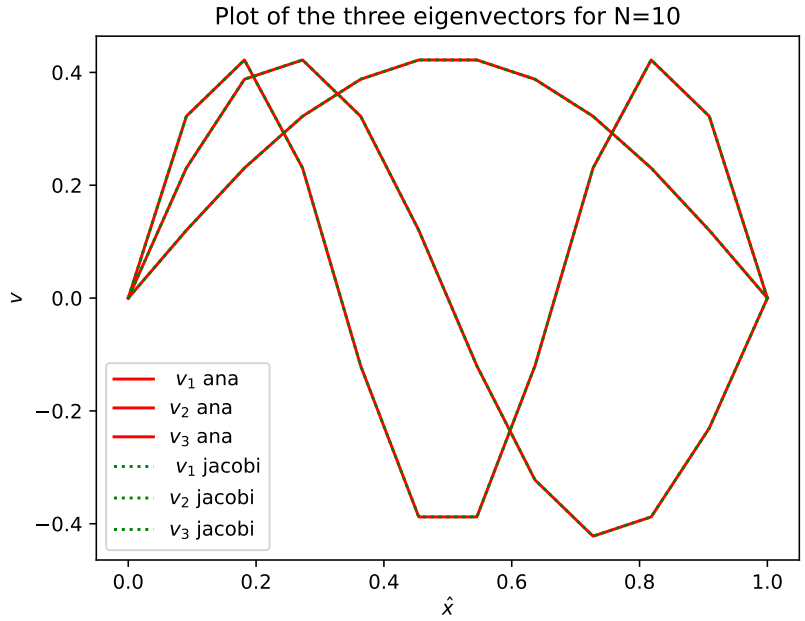
We have seen that to solve the differential equation is to find the eigenvalues and vectors for a tridiagonal matrix, where the size of the matrix N is the resolution of the equation.

We can solve the differential equation and plot the eigenvectors for the three smallest eigenvalues. The result for $N = 10$ can be found in figure(3). The results look like the first three modes of an oscillation, but has quite a low resolution. We can compare this with the analytical eigenvectors, as seen in figure(4). Here we see that they overlap. This is to be expected since we defined ϵ to be 10^{-3} meaning that the error should be smaller than this.

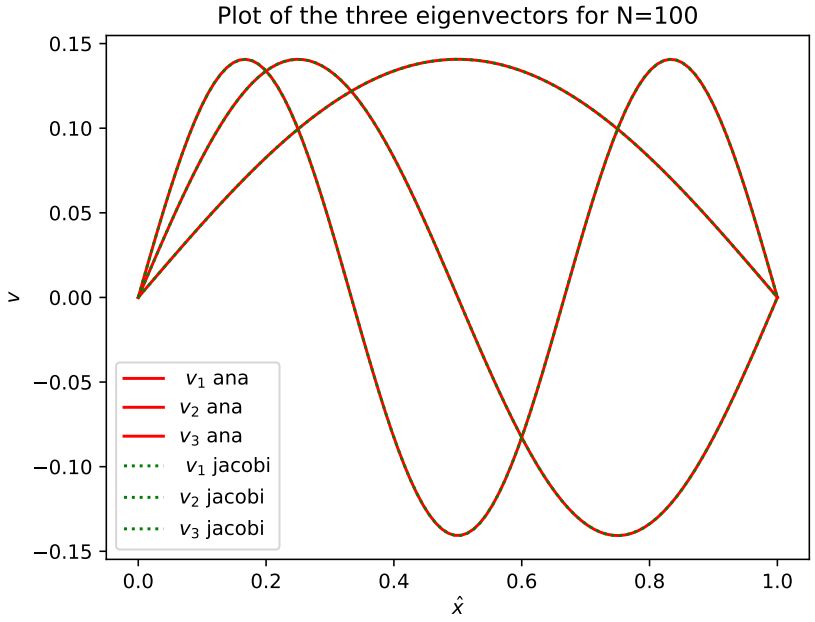
In figure(5) we see the same for $N = 100$. Here we have a better resolution of the solution, and the eigenvectors are still close to the analytical eigenvectors.



Figur 3. Plot of the three eigenvectors (v_1, v_2 and v_3) corresponding to the three lowest eigenvalue for $N = 10$.



Figur 4. Plot of the three eigenvectors (v_1, v_2 and v_3) found with analytical and the ones found with the Jacobi method. We see that the eigenvectors are almost equal, even if they are calculated in a completely different manner.



Figur 5. Plot of the three eigenvectors corresponding to the three lowest eigenvalues for the matrix size of $N = 100$. Three of them are calculated analytically and three of them calculated with the Jacobi algorithm, and we see that they also closely coincide.

---

[1] FYS3150 teaching team. Project 2. https://anderkve.github.io/FYS3150/book/projects/project2.html. [Online; accessed 27-September-2021].
[2] Morten Hjorth-Jensen. *Computational Physics*. 2015.