Project 2 FYS4150

Vetle Vikenes, Johan Mylius Kroken & Nanna Bryne
(Dated: September 27, 2022)

The code is available on GitHUb at https://github.com/Vikenes/FYS4150.

INTRODUCTION

To describe a one-dimensional buckling beam, we have the second order differential equation

$$\gamma \frac{\mathrm{d}u}{\mathrm{d}x} = -Fu(x), \quad x \in [0, L] \tag{1}$$

with $u(0) = u(L) = 0$. ...

PROBLEM 1

We define $\hat{x} \equiv x/L$. Now $\mathrm{d}^2\hat{x}/\mathrm{d}x^2 = L^{-2}$ and we can rewrite eq. (1).

$$\begin{aligned}
\gamma \frac{\mathrm{d}^2u}{\mathrm{d}\hat{x}^2}\frac{\mathrm{d}\hat{x}}{\mathrm{d}x} &= -Fu(x) \\
\frac{\gamma}{L^2}\frac{\mathrm{d}^2u}{\mathrm{d}\hat{x}^2} &= -Fu(\hat{x}) \\
\frac{\mathrm{d}^2u}{\mathrm{d}\hat{x}^2} &= -\frac{FL^2}{\gamma}
\end{aligned}$$

Letting $\lambda \equiv FL^2/\gamma$ yields

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

PROBLEM 2

In order to make sure that we can set up the tridiagonal $N \times N$ matrix A correctly, we write a short program in C++ that (1) defines A for $N = 6$, (2) solves $A\mathbf{v} = \lambda\mathbf{v}$ using the Armadillo library and (3) compares the solution to the analyical result.

PROBLEM 3

a)

In C++, we write a function that identifies the largest off-diagonal element in absolute value in a symmetric Armadillo matrix and notes the matrix indices (in the upper tiangle) of this element.

b)

We test the function on the matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & -0.7 & 0 \\ 0 & -0.7 & 1 & 0 \\ 0.5 & 0 & 0 & 1 \end{pmatrix}.$$

In particular, we make sure that the C++-function yields the maximum value 0.7 at row $k = 1$ and column $l = 2$.

PROBLEM 4

a)

In order to implement the Jacobi rotation algorithm we write two separate algorithms: Jacobi rotate, and Jacobi eigensolver:

Algorithm 1 Jacobi rotation

if $A_{kl}^m = 0$ **then**
  $c = 1$
  $s = 0$
  $t = 0$
else
  $\tau = (A_{ll}^m - A_{kk}^m)/(2A_{kl}^m)$
  if $\tau > 0$ **then**
    $t = 1/(\tau + \sqrt{1+\tau^2})$
  else
    $t = -1/(-\tau + \sqrt{1+\tau^2})$
  $c = 1/(\sqrt{1+t^2})$
  $s = ct$
$A_{kk}^{m+1} = c^2 A_{kk}^m - 2csA_{kl}^m + s^2 A_{ll}^m$
$A_{ll}^{m+1} = c^2 A_{ll}^m + 2csA_{kl}^m + s^2 A_{kk}^m$
$A_{kl}^{m+1} = 0$
$A_{lk}^{m+1} = 0$
for $i = 0, 1, 2, \ldots, N-1$ **do**
  if $i \neq k \wedge i \neq l$ **then**
    $A_{ik}^{m+1} = cA_{ik}^m - sA_{il}$
    $A_{ki}^{m+1} = A_{ik}^{m+1}$
    $A_{il}^{m+1} = cA_{il}^m + sA_{ik}^m$
    $A_{li}^{m+1} = A_{il}^{m+1}$
  $R_{ik}^{m+1} = cR_{ik} - sR_{il}$
  $R_{il}^{m+1} = cR_{il} + sR_{ik}$

b)

PROBLEM 5

a)

Consider our symmetric, tridiagonal matrix $A \in \mathbb{R}^{N\times N}$ with signature (a, d, a). Let M denote the number of transformations needed for a Jacobi rotation algorithm to converge. That is, M represents the number of iterations in the Jacobi rotation algorithm needed for the transformed matrix A' to be similar enough[1] to a diagonal matrix.

For $N = 2, 3, \ldots, 100$, we run the Jacobi eigensolver and save the corresponding integer M. The result is plotted in Figure 1.

[1] Choosing the number $\varepsilon = 10^{-8}$ to be *close enough* to zero.

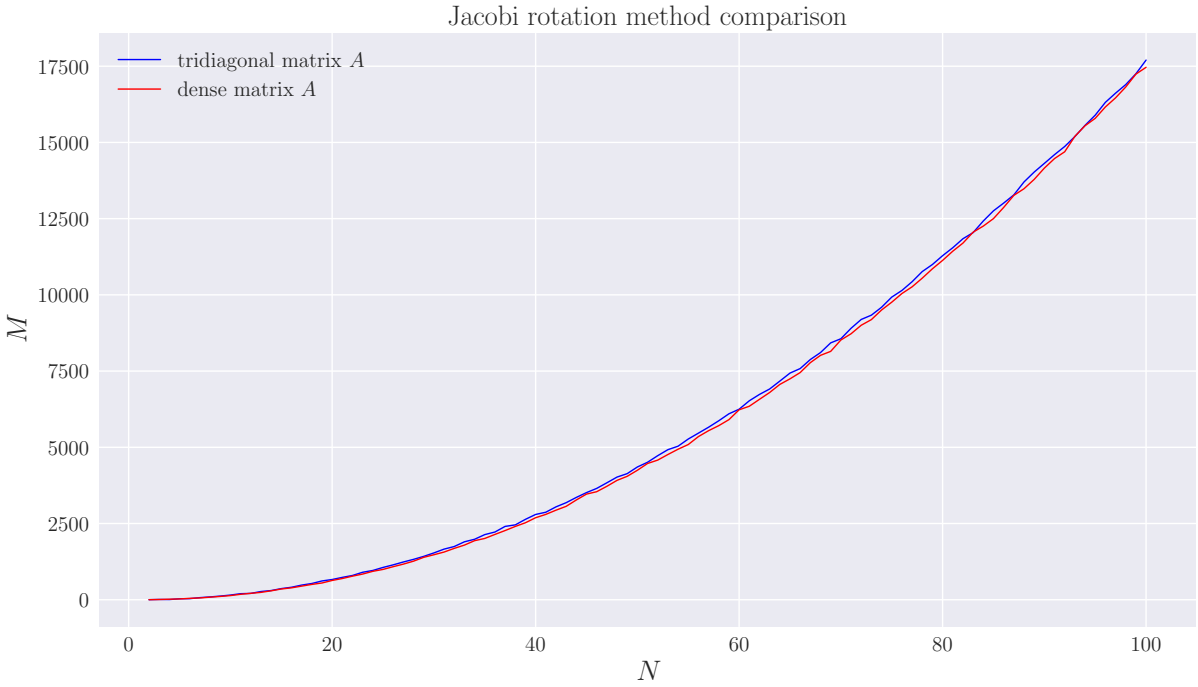

FIG. 1. The number of iterations M as function of the matrix size N. NEED UPDATE IN PLOT.

b)

PROBLEM 6

a)

For $n = 10$ steps, i.e. $n + 1 = 11$ points $\hat{x}_i$ and $A \in \mathbb{R}^{(n-1)\times(n-1)} = \mathbb{R}^{9\times 9}$, we solve the eigenvalue problem $A\mathbf{v} = \lambda\mathbf{v}$ using the Jacobi eigensolver. In addition, we solve the same problem with the analytic expressions for $\lambda^{(i)}$ and $\mathbf{v}^{(i)}$. This yields two versions of the normalised vectors $\mathbf{v}^{(i)}$, and for some i's these are counter-oriented. When we encounter this situation, the issue is solved by forcing the result from the Jacobi algorithm $\mathbf{v}^{(i)} \to -\mathbf{v}^{(i)}$.

The three resulting eigenvectors $\mathbf{v}^{(1)}$, $\mathbf{v}^{(2)}$ and $\mathbf{v}^{(3)}$ corresponding to eigenvalues $\lambda^{(1)}$, $\lambda^{(2)}$ and $\lambda^{(3)}$ s.t. $\lambda^{(i)} < \lambda^{(j)}$ for $i < j$, are plotted in Figure 2. The vectors are extrented with the boundary points, i.e. $v_0^{(i)} = v_0 = 0$ and $v_n^{(i)} = v_n = 0$ for all i.

b)

We do exactly the same as in **a)**, only now we use $n = 100$ discretisation steps. The resulting plot is presented in Figure 3.

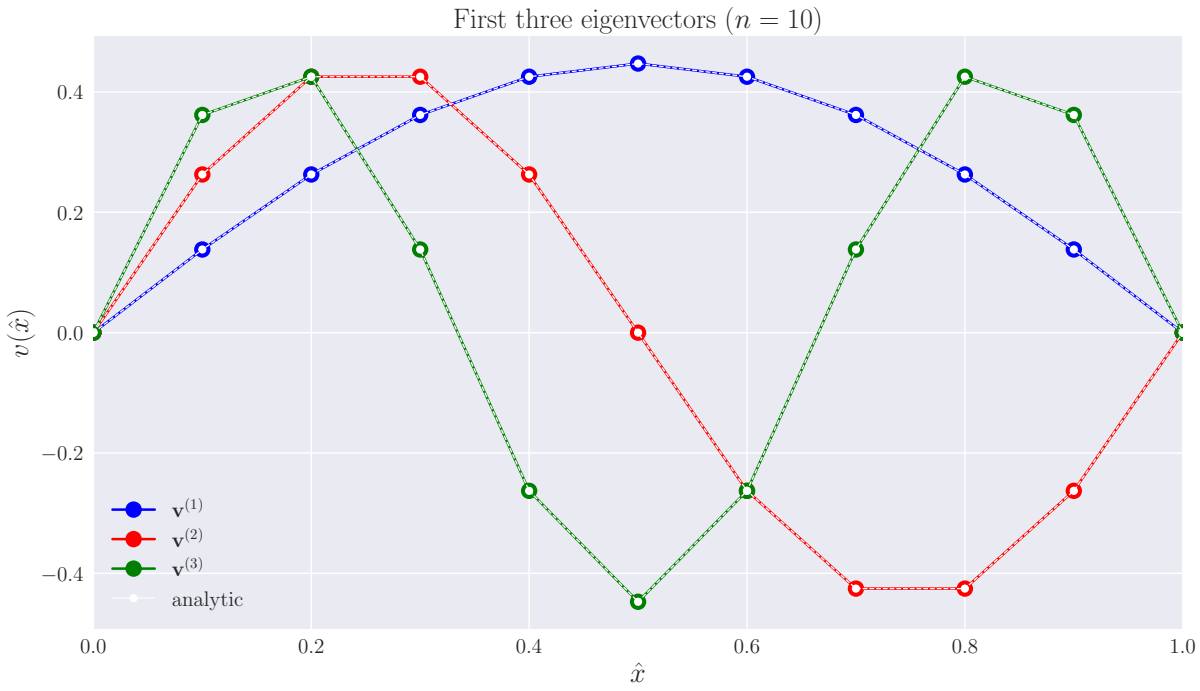

FIG. 2. The first three eigenvectors $\mathbf{v}^{(i)}$ respectively corresponding to the three lowest eigenvalues $\lambda^{(i)}$ computed with the Jacobi eigensolver using $n = 10$ discretisation steps. The white overplotted graphs are the predictions from the analytic expression.

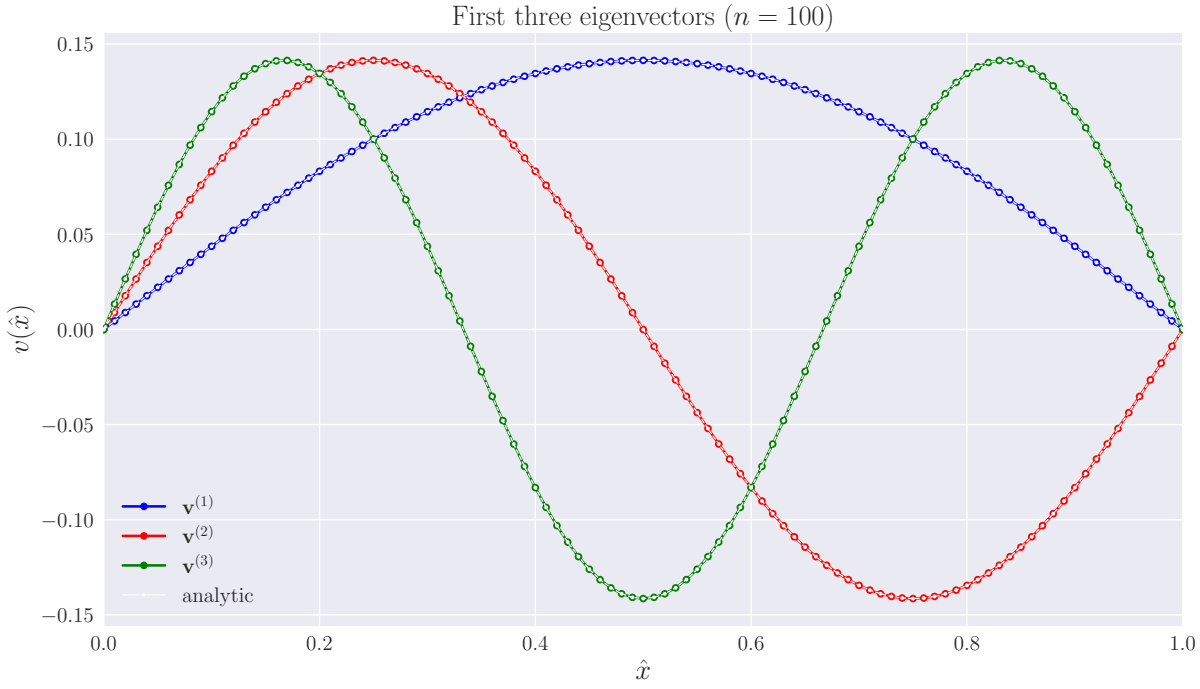

FIG. 3. The first three eigenvectors $\mathbf{v}^{(i)}$ respectively corresponding to the three lowest eigenvalues $\lambda^{(i)}$ computed with the Jacobi eigensolver using $n = 100$ discretisation steps. The white overplotted graphs are the predictions from the analytic expression.