FYS4150 - Project 2

github.com/isakrukan/FYS4150/Project2

Isak Cecil Onsager Rukan and Magnus Axelsen
(Dated: September 24, 2025)

I. PROBLEM 1

Consider a time-dependent horizontal beam of length L and $u(x)$ the displacement of the beam at position $x \in [0, L]$. A force is applied at $x = L$ directed towards $x = 0$. The beam is fastened, and so $u(0) = u(L) = 0$, but is allowed to rotate: $u'(x) \neq 0$.

This situation can be described by the second-order differential equation

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

where γ is some constant. Introducing $\hat{x} = x/L$ means that $d\hat{x}^2 = dx^2/L^2$, and thus

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

where we have defined $\lambda = FL^2/\gamma$.

II. PROBLEM 2

In `src/triDag.cpp` a tridiagonal matrix is implemented as a class that provides a method `compute_eigenvalues` which computes the eigenvalues and eigenvectors of the matrix. In `tests/test.cpp`, we run `test_TriDag`, which tests that the correct eigenvalues and eigenvectors are computed for a tridiagonal matrix of size $N = 6$.

III. PROBLEM 3

In `src/utils.cpp` we have created a function called `max_offdiag_symmetric`, which takes in a matrix A and two integers, k and l. This function assumes that A is a symmetric matrix, and returns the greatest off-diagonal element of A (in absolute value) and updates k and l by reference to be the row and column indices of this element, respectively. We test that this is implemented correctly using

$$A = \begin{pmatrix} 1 & 0 & 0 & 0.5 \\ 0 & 1 & -0.7 & 0 \\ 0 & -0.7 & 1 & 0 \\ 0.5 & 0 & 0 & 1, \end{pmatrix} \tag{3}$$

for which `max_offdiag_symmetric` should return -0.7 and set $(k, l) = (2, 1)$.

IV. PROBLEM 4

a)

The Jacobi algorithm is implemented in `src/jacobi_eigensolver.cpp`. The program is broken down into a function `jacobi_rotate()` performing one single Jacobi rotation, and a function `jacobi_eigensolver()` repeatedly calling `jacobi_rotate()` until the matrix is diagonal within some tolerance.

b)

A test function `test_jacobi()` is implemented in `tests/test.cpp`. The function constructs a 6×6 tridiagonal matrix and compares the eigenvalues and eigenvectors computed by the Jacobi algorithm to the analytic solutions.

V. PROBLEM 5

a)

We apply the Jacobi algorithm for different matrix sizes N, and count the number of transformations required for the algorithm to converge. Assuming the the number of required iterations roughly follows a power law $O(N^k)$, one can estimate the exponent k by linear regression on a log-log scale. The results are shown in figure 1, and we find that $k \approx 2.06$.

b)

Whereas a tridiagonal matrix starts with $O(N)$ off-diagonal elements, the corresponding number for a dense matrix will be $O(N^2)$. Given that the tridiagonal case scales roughly as $O(N^2)$, one may therefore expect that the additional factor of N in the number of off-diagonal elements leads to an additional factor of N in the number of required transformations. That is, one expects the dense case to scale as $O(N^3)$.

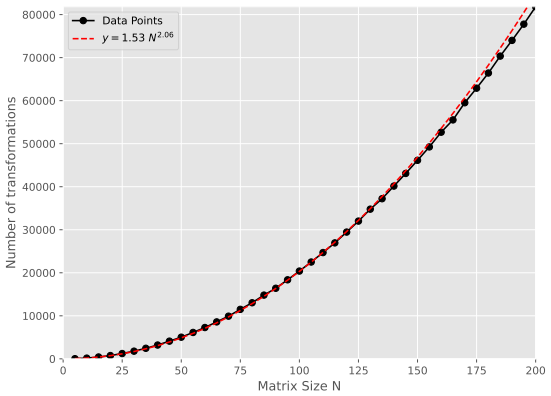

FIG. 1. Number of transformations required for the Jacobi algorithm to converge for different matrix sizes N.

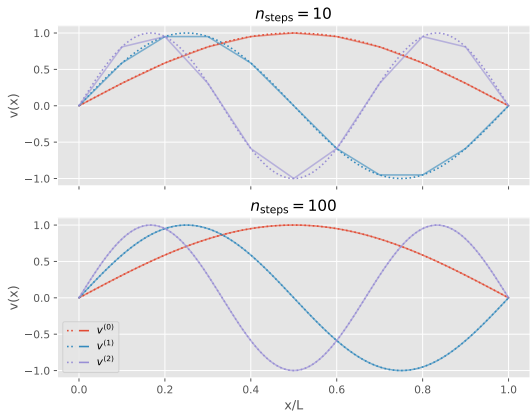

FIG. 2. The three lowest eigenmodes of the beam, for different numbers of discretization steps. Numerical solutions are given by the solid lines, with corresponding analytic solutions given by the dashed lines.

VI. PROBLEM 6

The three lowest eigenmodes of the beam, for different numbers of discretization steps, are shown in figure 2. The analytic solutions are also included for comparison. Notably, the numerical solutions are quite accurate even when the number of discretization steps is relatively small.