Project 2
For the course FYS3150

Erik Grammeltvedt, Erlend Tiberg North and Alexandra Jahr Kolstad

September 28, 2019
Week 37 - 40

Contents

1 Abstract

bla bla bla bla bla bla bla

2 Introduction

All programs are found at our GitHub-repository.

3 Method

Hvilke filer som er til hvilke oppgaver

3.1 Exercise a)

In this exercise we are going to prove that $\vec{w}_i = U\vec{v}_i$ is an orthogonal or unitary transformation that preserves the dot product and orthogonality. We start by multiplying $\vec{w}_j^T$ with $\vec{w}_i$ to take the vector product, also called the dot product. If the vector product of these vectors is equal to δ_{ij}, given by $\vec{v}_j^T \vec{v}_i = \delta_{ij}$ in the exercise, then the dot product and orthogonality is preserved. In this exercise we assume that $U^T U = I$, where I is the identity matrix, because this defines a unitary matrix U which we compute with in this exercise.

The vector product is calculated as followed:

$$\begin{aligned}
\vec{w}_j^T \vec{w}_i &= (U\vec{v})^T U \vec{v}_i \\
&= \vec{v}_j^T U^T U \vec{v}_i \\
&= \vec{v}_j^T \vec{v}_i \\
&= \delta_{ij}
\end{aligned}$$

The vector product of $\vec{w}_j^T$ and $\vec{w}_i$ is δ_{ij}, which proves that the dot product and orthogonality is preserved for the transformation.

3.2 Exercise b)

3.2.1 Calculations

In this prject we compute with a symmetric matrix, similar to the matrix $\mathbf{A}$ in project 1. This matrix is given by the matrix equation

$$
\begin{bmatrix}
d & a & 0 & \dots & 0 & 0 \\
a & d & a & \dots & 0 & 0 \\
0 & a & d & \dots & 0 & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & 0 & a & d & a \\
0 & 0 & 0 & 0 & a & d
\end{bmatrix}
\begin{bmatrix}
u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1}
\end{bmatrix}
= \lambda
\begin{bmatrix}
u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_{N-2} \\ u_{N-1}
\end{bmatrix}
$$

where $d = \frac{2}{h^2}$ and $a = -\frac{1}{h^2}$. **still correct??**

λ are eigenvalues given by the equation

$$
\lambda_j = d + 2a \cos\left(\frac{j\pi}{N+1}\right)
$$

given for $j = 1, 2, ..., N$.
Skal kommentere noe for egenverdi og egenvektor?

3.2.2 The programming

Commenting the code `jacobimethod.cpp`. OBS.

The program starts with defining a function for finding the max values of the off-diagonal elements. This is the function `offdiag`, which is taken from the lecture notes. The same applies for the function `Jacobi_rotate`. `Jacobi_rotate` is the function for rotating and computing the matrix. It calculates the equation

$$
\mathbf{B} = \mathbf{S}^T A \mathbf{S}
$$

for finding the diagonal matrix with the eigenvalues of the matrix $\mathbf{A}$. It also computes the eigenvectors, and stores them in a matrix $\mathbf{R}$.

In the main function we define the matrix $\mathbf{A}$ given by the constants d and a. See 3.2.1 for the definitions of the matrix. Then the clock is set to start, and it times how long the armadillo

Skal egentlig kommentere dette?????

The number of similarity transformations needed to reach the desired matrix depends on the dimension n. For instance a run of our matrix $\mathbf{A}$ given as a (10×10) matrix, there are 154 transformations needed. This number is only exact for this specific run, as it will change for any changes to the matrix, both size and elements.

In the lecture notes it states that for the Jacobi method there is no way to predict the number of transformations needed. See this file under *Discussion for Householder's method for eigenvalues*. **Riktig??**

Må vente litt med å kommentere dette
Koden bruker lengre tid både fordi matrisen er større, slik at det er mer å rotere og fordi den må utføre flere iterasjoner.

3.3 Exercise c)

3.3.1 Calculations

3.3.2 The programming

3.4 Exercise d)

3.4.1 Calculations

3.4.2 The programming

3.5 Exercise e)

3.5.1 Calculations

3.5.2 The programming

4 Results and discussion

Our results are as shown in the Appendix. We also have `.txt`-files for all the raw data generated by the projects up on GitHub.

5 Conclusion and perspective

6 Appendix

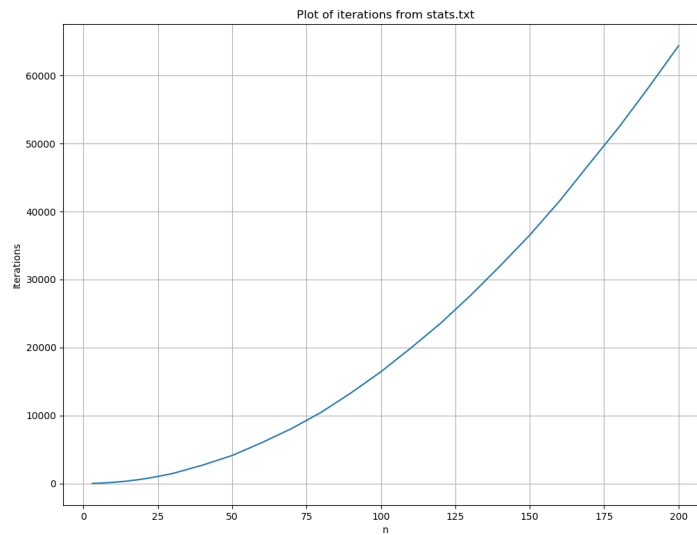

Figure 1: The plot of iterations for the Jacobi method as function of the dimension n of the matrix $\mathbf{A}$.
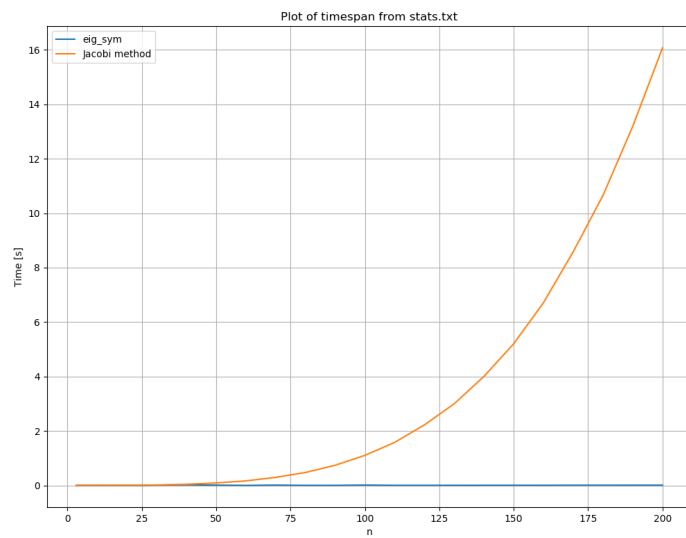
Figure 2: The plot of the time the function `eig_sym` from Armadillo uses and the time Jacobi method uses as functions of the dimension n if the matrix $\mathbf{A}$.

7 References

Link to the PDF for Project 2.

Our GitHub-repository.

Link to lecture slides in FYS3150 - Computational Physics.