Final Exam: Inverse iteration algorithm for eigenvalues

Simon Vendelbo Bylling Jensen*†
Stud. Nr. 201507956
Au Id. Au545766

June 6, 2019

Abstract

The solution of exercise 10 for the final exam. An implementation f an inverse iteration algorithm for eigenvalues and eigenvectors. Such an implementation has been made, the first relying on a Golub-Kahan-Lanczos bidiagonalization iterating using a calculated inverse of the matrix, as well as a method using QR-decomposition by modified Gram-Schmidt orthogonalization to iterate through the solution of a linear system using backsubstitution.

Introduction and Problem description

I have recieved exercise number 10, since my student number 56 by modulus 23 will give 10. The problem is to implement a variant of the inverse iteration method that calculates the eigenvalue closest to a given number s and the corresponding eigenvector. The problem of finding eigenvalues and corresponding eigenvectors are crucial to various fields of physics and engineering. Especially for higly complicated quantum systems, complicated enginerring systems and much more, the task of finding an individual eigenvalue and determine its corresponding eigenvectors is essential. The algorithm, which will be demonstrated is a fast and effective way of doing excactly that.

At first the implementation and theory of inverse iteration method the will be described in section 1 and afterwards the different related methods will be compared and analysed in section **??**. Implementations and simple calculations using the examined methods will similarly be demonstrated in the main.c file, and which results can be found in the out.txt file, as well as described hereafter.

*AU E-mail: 201507956@uni.au.dk
†CERN E-mail: simon.vendelbo.jensen@cern.ch

1 Implementation and Theory

The implementation and theory all relies on the lecture notes from ref. [1, Chapter 12 - Eigenvalues 2: Power methods and Krylov subspace methods]. The refined inverse iteration method relies on a few simple principles. All of these principles will be described by different examples, which are all demonstrated in the main.c file, and which results can be found in the out.txt file. The description of the inverse iteration method will thereby be clear, by going through simpler implementations of the same procedure. This will be the case for sections **??**. Hereafter first the implementation using a Golub-Kahan-Lanczos bidiagonalization will be described in section **??** at at last, the refined incerse iteration method will be described in section **??**.

1.1 The Power Method

Since the inverse iteration method is a cleverly designed power method, it is essential to introduce the basics of the power methods. The simplest power method is the power iteration which works by multiplying a random vector $\vec{V}$ on a matrix A of similar

size, which we want to know the largest eigenvalue and corresponding eigenvector. By continiously multiplying the vector $\vec{V}$ on the matrix A, through the iteration

$$\vec{V}_{i+1} = A\vec{V}_i \tag{1}$$

we are able to aproximate the largest eigenvector and largest eigenvalue very cleverly. The vector will at some point simply converge to the largest eigenvector. This is since by the multiplication, the dimension in which the eigenvector corresponding to the numerically largest eigenvalue will give the largest enhancement of the vector, $\vec{V}$, and for each iteration, this will have positive feedback and will at some point have $\vec{V}$ converge to some vector, which normalized will approximate that eigenvector. If the vector $\vec{V}$ will converge to the eigenvector, then at some point every iteration and thereby multiplicaltion will fullfill the eigenvector identity

$$\vec{V}_\lambda A = \lambda A \tag{2}$$

We will thereby be able to approximate the largest eigenvalue continuously by finding the factor, in which V increases for every iteration. This parameter which are contiously detected are called the Rayleigh quotient, and will at one point converge to the largest eigenvalue. This is given as

$$\lambda\left[\vec{V}_i\right] = \frac{\vec{V}_i^T A\vec{V}}{\vec{V}_i \cdot \vec{V}_i} \tag{3}$$

For the implementation in main.c. the satisfaction critera for when an eigenvalue has been found, is when this Rayleigh quotient have converged and do not improve anymore with an iteration. This method is particularly effective in use, since it do not have to do any large matrix decompositions, which the computation of eigenvalues usually requires.

2 The Inverse Power method

Intuitively, when one would be able to converge to the numerical largest eigenvalue by continuisly multiplying with the matrix A one would expect, that a similar procedure could be done to find the numerically smallest eigenvalue. This is true, but the cost of this implementation, is the fact that one would in this case have to calculate the inverse of the matrix A. By the similar argument as for the power method, by multiplying with the inverse of A, the minimum eigenvalue, will now be dominant and the vector $\vec{V}$ will converge to the corresponding eigenvector. The iterations are hereby similar to equation (1) and given as

$$\vec{V}_{i+1} = A^{-1}\vec{V}_i \tag{4}$$

The inverse are for this case, for the implementation as found un

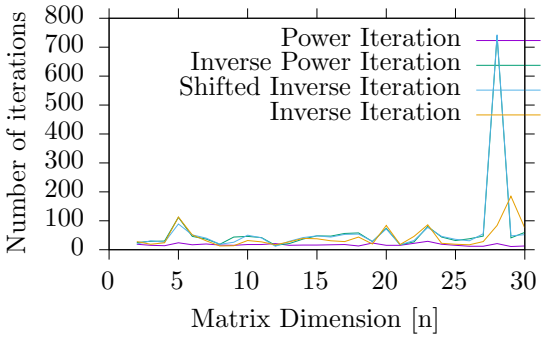

Figure 1: Comparison between the calculated arctangent function using the differential equation found in "myarctan.c" and the arctangent function from the math.h library.

References

[1] Fedorov, D.V. *Introduction to Numercal Methods*, Lecture Notes, Aarhus University, 2019.

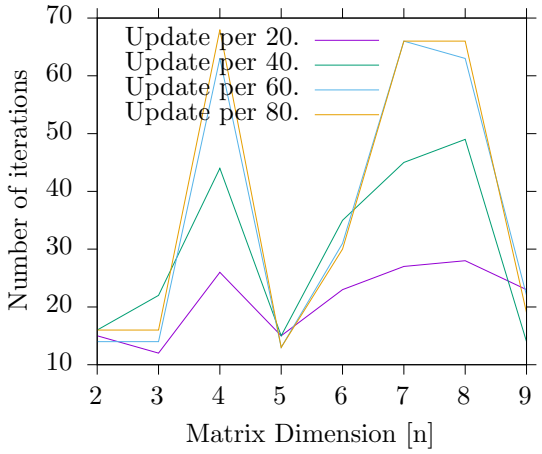

Figure 2: Comparison between the calculated arctangent function using the differential equation found in "myarctan.c" and the arctangent function from the math.h library.

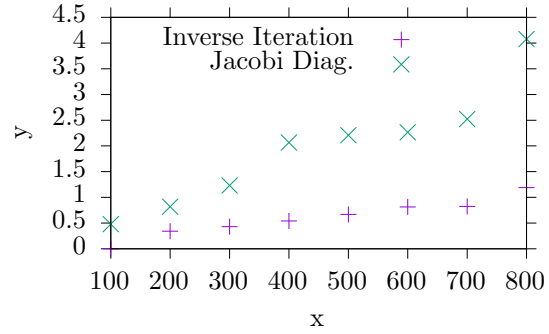

Figure 3: Examination of the computing time as a function of....