Final Exam: Inverse iteration algorithm for eigenvalues

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

June 7, 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

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.

1 Implementation and Theory

The implementation and theory all relies on the lecture notes from ref. [2, 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 1.3 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

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

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

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

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.

1.2 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, found using the Golub-Kahan-Lanczos bidiagonalization as prevously implemented in the Linear Equation exercise C. This procedure is described in the following subsection 1.2.1. When this is found, one can continiously apply itteration 4 to find the most dominant eigenvector when inverse iterating, corresponding to the eigenvector with the numerically smallet eigenvalue. This time since we apply the A^{-1} instead of A, the eigenvector identity now becomes

$$A^{-1}\vec{V}_\lambda = \frac{1}{\lambda}\vec{V}_\lambda \tag{5}$$

instead of equation (2). And therefore the Rayleigh parameter from (3) will not be approximating the lowest numerical eigenvalue, but instread its inverse.

1.2.1 Golub Kahan Lanczos bidiagonalization

The Goloub-Kahan-Lanczos bidiagonalization is the subject of the previous exam exercise 9, and the linear-equation exercise C, this therby been made to use in this implementation as well. The procedure of implementing this is described in ref. [**?**]. The method is usualy the start of a single value decomposition, and work by computing the unitary othogonal matricies U and V, such that

$$U^*AV = B, \tag{6}$$

where B is bidiagonal. This procedure is related the Gram-Schmidt procedure, since it starts with one vector in both V and U where it orthogonalize it to the previous vectors, it normalizes the vector, but furthermore than the simple Gram-Schmidt procedure, the Golub-Kahan-Lanczos bidiagonalization makes sure, through relations

$$\alpha_k \vec{U}_k = A\vec{V}_k - \beta_{k-1}\vec{U}_{k-1}, \tag{7}$$

$$\beta_k \vec{V}_{k+1} = A^* \vec{U}_k - \alpha_k \vec{V}_k, \tag{8}$$

with $\alpha_k = \vec{U}_k^* A \vec{V}_k$ and $\beta_k = \vec{U}_k^* A \vec{V}_{k+1}$ that, where the Gram-Schmidt procedure will return a triangular matrix, which can be solved for finding the inverse through backpropagation, the Golub-Kahan-Lanczos returns a matrix B for finding the inverse, that will be quicker to solve through backpropagation, since it is bidiagonal. The solution for the inverse, can thereby easily be found, by backsubstitution of the linear system

$$A \cdot A^{-1} = I, \tag{9}$$

$$UBV^T A^{-1} = I, \tag{10}$$

$$B\left(V^T A^{-1}\right) = U^T. \tag{11}$$

This linear system is then solved for $V^T A^{-1}$ which can then be multiplied with the orthogonal V for finding the inverse. Many other aspects of this algorithm could be widely covered, but since this is not the point of the exercise, we will settle with the fact that the Golub-Kahan-Lanczos bidiagonalization procedure are able to provide us with A^{-1} for the inverse power method.

1.3 Shifted Inverse Iteration

Since we in the final iteration method, will want to estimate a eigenvalue with corresponding eigenvector, which relies closely to a given value s which is not nessesarely the largest or the smallest eigenvalue. Then we need to implement a new method, that will converge to a wanted region of eigenvalues. This is done using the previous inverse iteration method, since it converges to a controlled value 0, which will then has to be shifted to the given point s This can be simply done with an understanding of the linear algebra of eigenvalues and eigenvectors. If one apply the shift

$$A \to A - sI \tag{12}$$

Any eigenvectors of A will on this shifted matrix still remain an eigenvector, but through equation (2) give

$$(A - sI)\,\vec{V}_\lambda = A\vec{V}_\lambda - sI\vec{V}_\lambda, \tag{13}$$

$$= \lambda\vec{V}_\lambda - s\vec{V}_\lambda, \tag{14}$$

$$= (\lambda - s)\,\vec{V}_\lambda. \tag{15}$$

Thereby introducing a shifted eigenvalue of $\lambda \to \lambda - s$. By introducing this shift, one can simply use the inverse power procedure with iterations as

$$\vec{V}_{i+1} = (A - sI)^{-1}\,\vec{V}_i \tag{16}$$

find any eigenvalues, with the expected eigenvalue area around s, in which this eigenvalue now will be shifted to be close to 0 for the inverse power method to converge to it. Afterwards one must remember to shift eigenvalue back by the same s, to find its actual location, whereas the eigenvector remains the same. This procedure has been implemented and tested in the main.c file using Golub-Kahan-Lanczos bidiagonalization method to find the inverse of the shifted $(A - sI)^{-1}$. A more refined way of doung the shifted inverse power method is the inverse iteration method, in which one do not need to calculate the inverse of A, but instead solves a linear system. This will now follow.

1.4 Inverse Iteration Method

The main difference in the inverse power method and this refined inverse iteration method is the change from iteration through multiplication as in equation (17), to now iterate through finding the next vector by solving the linear system

$$(A - sI)\,\vec{V}_{i+1} = \vec{V}_i \tag{17}$$

This could in fact be done by the previously defined Golub-Kahan-Lanczos bidiagonalization method, since this will end up in a backsubstitution, which is significantly simple for the bidiagonalized matrices. However to demonstrate another procedure of solving theese systems, this time a simple QR-decomposition by modified Gram-Schmidt orthogonalization has been made as in the linear equation exercise B. This method will be discussed in the following subsection, but differs from the Golub-Kahan-Lanczos bidiagonalization method by being faster to compute initially, since it only needs to construct a single orthogonal matric. But since the trangular matrix from the QR decomposition is slower to make a complete backsubstitution of, than the quick backsubstitution of the bidiagonalized matrix, then the main differnce between theese two methods for this implementation. Is the QR-decomposition being quicker to initialize, but more costly per iteration, whereas the Golub-Kahan-Lanczos method is more costly to initialize, but quicker to run. Since we want to implement an algorithm, that allows the user to specify how often the system should replace the user provided s with the estimated eigenvalue from the Rayleigh quotient, whereas the QR or Golub-Kahan-Lanczos method would be reinitialized, then for a demonstration of an inverse iteration algorithm with many continious updates, the QR-decomposition will be the fastest, and for one in which the user-supplied estimate is very well, and one would not need to replace it, with the estimate from the Rayleigh quotient, one would prefer the Golub-Kahan-Lanczos method.

In reference [2] it states that in practise the method is usually used for finding a result when a good approximation for the eigenvalue is known, with only a few iterations. If only a few iterations shall be made, then the fast initializing QR-decomposition will be the fastest, and therefore the one i will choose for the iteration procedure.

The QR-decomposition will be described in the following section **??**, and will through backsubstitution determine the new vector at each iteration. With the optimized iteration step, the rest of the inverse iteration method is simple implemented as for the shifted inverse power method as described in section 1.2.

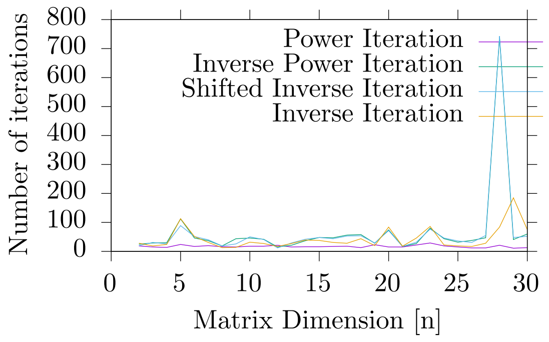

Figure 1: Comparison between the calculated arctangent function using the differential equation found in "myarctan.c" and the arctangent function from the math.h library.

1.4.1 QR-Decomposition by modified Gram-Schmidt orthogonalization

References

[1] Dongarra, J. *Golub-Kahan-Lanczos Bidiagonalization Procedure.*, Netbib.org/..., 2019.

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

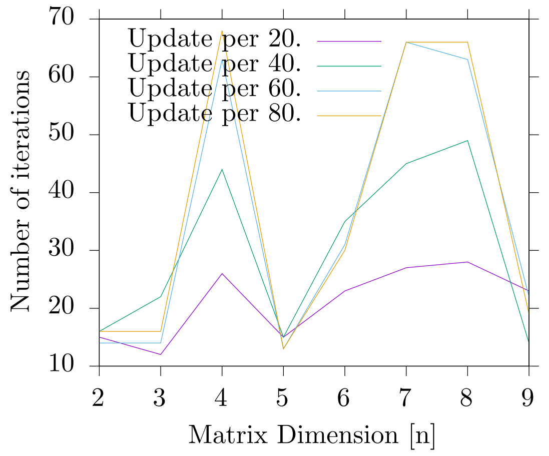

Figure 2: Comparison between the calculated arctangent function using the differential equation found in "myarctan.c" and the arctangent function from the math.h library.

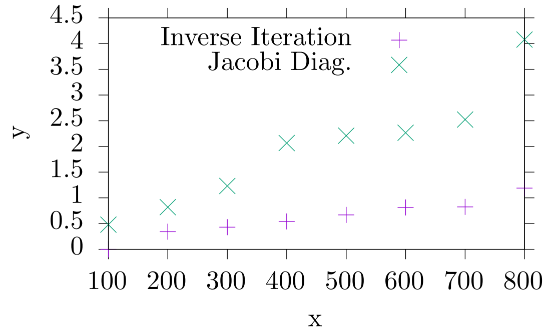

Figure 3: Examination of the computing time as a function of....