Eötvös Loránd University
Faculty of Science

Emese Szakály

Numerical Computations in Linear Algebra

BSc Thesis in Applied Mathemathics

Supervisor:

Alice Fialowski
Associate professor

Department of Algebra and Number Theory

Budapest, 2019

Acknowledgements

First of all, I would like to thank my supervisor, Alice Fialowski, for the help she gave, the positive feedback during our meetings and the support of my ideas. I could learn how to accomplish a bigger project on my own for the future.

I am grateful to my Erasmus studies, because that opened my eyes to numerical linear algebra. The courses I took motivated me a lot to dig deeper into this topic.

I appreciate all the support I got from my parents during these months, including the delicious meals that gave me energy. I am happy that we could share our university experiences with Kincső, also in this period. We made it through the hard times, helping each other, studying together, and having a lot of fun.

Contents

1 Introduction

Problems of linear algebra need to be solved every day in different areas of science and engineering. When they can not be solved analytically, numerical methods come into prominence. They play a really important role in life, e.g. when building airplanes, bridges, doing operations and machine learning. For this reason it is essential to see in which case how effective and fast is an algorithm.

In subsection *2.1* numerical methods are introduced for solving systems of linear equations $A\mathbf{x} = \mathbf{b}$. We will see how Gauss elimination can be improved and made a really effective tool. Knowing the computational costs of a modification it will be discussed if it is worth to be used or not. Subsection *2.2* elaborates what it means if a system is ill-conditioned, and why it is so dangerous while using numerical methods. Furthermore, we will see two methods from recent articles showing ideas how to increase accuracy when the system is ill-conditioned.

Section *3* presents four different methods for solving nonlinear equations. The first two, Bisection and Regula falsi are bracketing methods. They keep the solution in the start interval, which is reduced in every step until it is small enough. Newton's and Secant methods start with an initial guess, and with using a scheme they produce better an better solutions. The algorithms are compared in the sense of efficiency and running time.

Section *4* details the topic of eigenvalues. First we will get familiar with the Gershgorin discs (*4.1*), which gives a spectacular way of approximating eigenvalues. In subsection *4.2* Power iteration and its variants show how to compute one eigenvalue-eigenvector pair of the given matrix. Its improved version is the orthogonal iteration, which can compute more eigenvalues at the same time. Last but not not least, the QR method is presented, which finds all the eigenvalues of the matrix. For this reason, this is the most widely used method for computing eigenvalues. In the last century it was listed as one of the top 10 algorithms. We will see what is the Singular Value Decomposition (*4.3*) and how it is applied to approximate matrices. One article is presented here which solves ill-conditioned systems with the use of SVD. Spectrum slicing (*4.4*) is a way of finding one or more eigenvalues of a symmetric matrix. For this we need the Cauchy interlacing theorem, Sturm sequence and the previously mentioned Bisection method.

The Appendix contains the code of seven algorithms discussed in the thesis.

Apart from a few cases, the books and articles I used are mentioned in the beginning of each section. Beside the material listed in the References I utilized a lot from what I learnt in Eindhoven during my Erasmus semester.

2 Systems of linear equations

The problem of solving many linear equations simultaneously arises whenever we have multiple variables that are dependent on each other. In these cases we are interested in solving the system of linear equations

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{n\times n}, \quad \mathbf{b} \in \mathbb{R}^n.$$

Since it appears in several areas, such as science and engineering, it is essential to find accurate and efficient ways of solving these problems. Of course, for 2-3 variables one could solve it manually, but for more than 3 unknowns (or equations) the problem becomes much more complicated.

2.1 Numerical methods for solving systems of linear equations

For writing this subsection I used the following books: [3], [7], [11].

Here we consider only direct numerical methods, which means that the solution is calculated by performing arithmetic operations with the equations. The aim is to transform the initially given system to an equivalent system of equations that can be solved easily. These preferred forms are upper triangular, lower triangular and diagonal matrices. One can obtain the solution of the upper and lower triangular matrices with backward and forward substitution, respectively. The diagonal matrix contains the solution in its diagonal entries.

2.1.1 Gauss elimination method

The most known method for solving systems of linear equations is Gauss elimination. The point is to transform the given coefficient matrix A to an upper triangular form, so that we can easily solve this equivalent system by backward substitution. We start from the first row, and continue until the second last row. In each step we have a pivot element (a_{ii}), and we eliminate all the elements below the pivot. We execute this by subtracting the pivot i^{th} row (row of the pivotal element) from the j^{th} (j=i+1,...n) rows $m_{ij} = \frac{a_{ji}}{a_{ii}}$ times, so that the elements a_{ji}, (j=i+1,...n) become 0.

The code for Gauss elimination can be found in the Appendix *A.1.*

Operations count

Gauss elimination with back substitution for an $n \times n$ system requires $\frac{n^3}{3} + n^2 - \frac{n}{3}$ multiplications/divisions, and $\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6}$ additions/subtraction. As n grows, $\frac{n^3}{3}$ will be the leading term in both of these expressions.

Weak points of the algorithm

- If the pivot element is 0, the algorithm cannot proceed with the division when we try to eliminate the elements below the pivot.
- If the magnitude of the pivot element is small relative to the other terms in the pivot row, it leads to bigger round-off errors.

2.1.2 Gauss elimination with pivoting

Now the solution will be presented for the previously mentioned potential difficulties when applying Gauss elimination. To solve the problem when the pivot element is 0, we can switch rows. If there is a solution for the system, there is always a nonzero element below the 0 pivot element, and these two rows can be changed. If we take a look at the main step of the procedure, we can see that any round-off error is amplified by m_{ij}. Therefore, we would like to ensure that this multiplier is small. This can be accomplished by row interchanges. When eliminating elements in column j, we seek the largest element in column j, on or below the main diagonal, and then interchanging that element's row with row j. This is called **partial pivoting**. We reached that the multiplier will be smaller than one. Furthermore, pivoting in this manner requires $O(n^2)$ comparisons to determine the appropriate row interchanges, which extra expense is negligible compared to the overall cost of Gauss elimination, and therefore is outweighed by the potential reduction in round-off error.

Even though we avoided big round-off errors, a loss of significant digits can still appear, if the coefficients in the pivot equation are much larger or smaller than in the other equation. This can be fixed by scaling, which means that in every row the maximum absolute value of magnitude equals to 1. For this reason, we search in every row the element with the largest absolute value, and divide the equation by this number. It again adds only $O(n^2)$ comparisons to compute the scale factors, so it does not add a significant overhead to Gauss elimination. Gauss elimination with scaling and partial pivoting is an extremely effective tool.

The code for Gauss elimination with partial pivoting and scaling can be found in the Appendix *A.2*.

There is an extension of partial pivoting, called **complete pivoting**. Here we compare all the elements in the matrix that are to the right and down from the pivoting position. We find the one with largest absolute value, and with row and column interchanges we move it to the pivoting position. This requires $O(n^3)$ comparisons, so it increases the computational costs quite a lot. Obviously, it is at least as effective as partial pivoting. Furthermore, in some special cases, this method can be more effective in controlling the effects of round-off errors than partial pivoting. Also, complete pivoting makes Gauss elimination stable, which means that the entries of the matrix will not grow exponentially when they are updated by elementary row operations. However, in reality, we rarely encounter with those special kind of matrices. For this reason complete pivoting is not used a lot in practice.

2.1.3 Gauss-Jordan elimination method

As a result of the Gauss-Jordan elimination we obtain a diagonal matrix with normalized elements in the diagonal. We reach this by dividing every row with its pivot element (so the pivot elements are equal to 1), and we eliminate all the elements below and above the pivot element, so finally A is transformed to the identity matrix, and the new vector **b'** is the solution.

The advantage of this algorithm is that we can solve the system for several right hand side vectors **b** at the same time by augmenting all the right hand side vectors to A.

Operations count

For an $n \times n$ system this method requires $\frac{n^3}{2} + \frac{n^2}{2}$ multiplications/divisions and $\frac{n^3}{2} - \frac{n}{2}$ additions/subtractions. So as n grows, $\frac{n^3}{2}$ is going to be the leading term in both expressions. We can see, that - especially for large matrices - Gauss-Jordan method requires much more operations than the Gauss elimination. Hence, practically it is not often used, but for example, for computing the inverse of a matrix it is a widespread method, since we can compute it with $B = I$, and the solution will be the inverse of A.

The code for Gauss-Jordan method can be found in the Appendix *A.3*.

2.1.4 LU decomposition

In the Gauss elimination procedure the elimination part requires much more operations than the back substitution. During the elimination both the matrix A and the right hand side vector $\mathbf{b}$ are changed, so it is not possible to solve the problem for more right hand side vectors simultaneously. We can see that it would be really useful to dissociate the operations made on A from the ones made on $\mathbf{b}$. One option for solving various systems of equations $A\mathbf{x} = \mathbf{b}$ that have the same coefficient matrices A but different constant vectors $\mathbf{b}$ is to first calculate the inverse of the matrix A. However, we know that the computational costs of calculating the inverse matrix are really high. Another way is to calculate the LU decomposition of A, where U is the upper triangular matrix which is the result of the Gauss elimination, and L is a lower triangular matrix with ones in its diagonal, and the m_{ij} multipliers used during the Gauss elimination are below the diagonal. Once we computed the LU factorization of A, we can easily solve the linear system $LU\mathbf{x} = \mathbf{b}$. First, we solve $L\mathbf{y} = \mathbf{b}$ by forward substitution, and then we solve $U\mathbf{x} = \mathbf{y}$ by backward substitution.

Operations count

This procedure uses $O(\frac{2}{3}n^3)$ for the matrix factorization and $O(n^2)$ for the forward and backward substitution.

This method is especially advantageous for solving systems that have the same coefficient matrices A but different constant vectors $\mathbf{b}$.

The code for the Gauss elimination using LU decomposition can be found in the Appendix *A.4*.

2.2 Ill-conditioned systems

For writing this subsection I used the following books [3], [7].

Consider the solution of the system

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{n\times n}, \quad \mathbf{b} \in \mathbb{R}^n. \tag{1}$$

First, recall some important matrix norms for $A \in \mathbb{R}^{n\times n}$.

Definition 2.1 (Induced matrix norm).
Let A be $\in \mathbb{R}^{n\times n}$. Then the induced matrix norm of A is

$$\|A\| = \max_{\mathbf{x}\in\mathbb{R}^n, \mathbf{x}\neq 0} \frac{\|A\mathbf{x}\|}{\|\mathbf{x}\|}$$

Properties of induced matrix norms:

- $\forall \mathbf{x} \in \mathbb{R}^n, A \in \mathbb{R}^{n\times n} : \|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|$
- All induced matrix norm satisfy $\|I\| = 1$
- Submultiplicity: $\|AB\| \leq \|A\|\|B\| \quad \forall A, B \in \mathbb{R}^{n\times n}$
- $\rho(A) \leq \|A\| \quad \forall A \in \mathbb{R}^{n\times n}$, where $\rho(A)$ is the eigenvalue of A which has the maximum absolute value.

Important induced matrix norms:

- 1-norm (column norm): $\|A\|_1 = \max_{j=1,2,\dots n} \sum_{i=1}^{n} |a_{ij}|$
- 2-norm (row norm): $\|A\|_2 = \sqrt{\rho(A^\mathsf{T}A)}$
- ∞-norm: $\|A\|_\infty = \max_{i=1,2,\dots n} \sum_{j=1}^{n} |a_{ij}|$

An example of noninduced matrix norms:

- Frobenius-norm : $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}^2}$

When trying to solve system (1) we are interested in how the solution $\mathbf{x}$ changes if the coefficient matrix A and the right hand side vector $\mathbf{b}$ have small changes. So what is the resulting error $\delta\mathbf{x}$ in $\mathbf{x}$ if A is perturbed by δA and $\mathbf{b}$ is perturbed by $\delta\mathbf{b}$?

Proposition 2.1.
The resulting relative error is

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A^{-1}\|\|A\|\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}\right).$$

Proof.
Consider the perturbed equality:

$$\begin{aligned} (A+\delta A)(\mathbf{x}+\delta\mathbf{x}) &= \mathbf{b}+\delta\mathbf{b} \\ \Leftrightarrow A\mathbf{x} + \delta A\mathbf{x} + A\delta\mathbf{x} + \delta A\delta\mathbf{x} &= \mathbf{b}+\delta\mathbf{b}. \end{aligned}$$

By neglecting the term $\delta A \delta \mathbf{x}$, and using $A\mathbf{x} = \mathbf{b}$ we find:

$$\delta A\mathbf{x} + A\delta\mathbf{x} \doteq \delta\mathbf{b}.$$

After arranging this equation we get that

$$\delta\mathbf{x} \doteq A^{-1}(-\delta A\mathbf{x} + \delta\mathbf{b}).$$

Taking the induced norm gives us the following:

$$\|\delta\mathbf{x}\| \leq \|A^{-1}\| \cdot (\|\delta A\| \cdot \|\mathbf{x}\| + \|\delta\mathbf{b}\|).$$

Since we are interested in the relative errors, divide the inequality by $\|\mathbf{x}\|$ to find

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A^{-1}\| \cdot \left(\|\delta A\| + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{x}\|}\right) = \|A^{-1}\| \cdot \|A\| \cdot \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\mathbf{b}\|}{\|A\| \cdot \|\mathbf{x}\|}\right).$$

By using $\|\mathbf{b}\| = \|A\mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$, we get

$$\frac{\|\delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \|A^{-1}\| \cdot \|A\| \cdot \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta\mathbf{b}\|}{\|\mathbf{b}\|}\right).$$

□

Definition 2.2 (Condition number of a matrix).
Let $A \in \mathbb{R}^{n\times n}$ be a nonsingular matrix. The condition number of A is defined as

$$\kappa(A) = \|A^{-1}\|\|A\|.$$

Proposition 2.2.
The condition number is always at least 1.

Proof.

$$\|A^{-1}\|\|A\| \geq \|A^{-1}A\| \geq 1.$$

□

Remark.
In practise the most important feature is that the condition number is a measure for singularity. If $\kappa(A)$ is close to 1, then the matrix is nearly singular, hence it is easy to invert. If $\kappa(A)$ is very large, then A is far from being singular. The following example illustrates that on the other hand, determinant is not always a good measure for singularity.

Example.
Let $\alpha \in \mathbb{R}$, $\alpha \geq 0$. Let $A = \alpha I$.
Clearly, A is nonsingular, $A^{-1} = \frac{1}{\alpha I}$.
Now we check what result the condition number and the determinant of A shows. $\kappa(A) = \|A^{-1}\|\|A\| = \|\alpha I \| \cdot \|\frac{1}{\alpha I}\| = \alpha \cdot \frac{1}{\alpha} = 1$. The condition number shows that A is nonsingular, therefore it proves to be a good measure for the singularity of A. $\det(A) = \alpha^n$. For $0 \leq \alpha \leq 1$ the determinant becomes very small, and for $\alpha \geq 1$ it becomes very large.

Theorem 2.3 (Perturbation theorem).
Let the matrix $A \in \mathbb{R}^{n\times n}$ be nonsingular. Let $B \in \mathbb{R}^{n\times n}$. Let $\|.\|$ be an induced matrix norm. Consider the perturbed matrix $A + B$.
If $\|BA^{-1}\| < 1$, then $A + B$ is nonsingular and

$$\frac{\|A^{-1}\|}{1 + \|BA^{-1}\|} \;\leq\; \|(A + B)^{-1}\| \;\leq\; \frac{\|A^{-1}\|}{1 - \|BA^{-1}\|}.$$

Proof.
Since $\rho(BA^{-1}) \leq \|BA^{-1}\| \leq 1$, it follows that $I + BA^{-1}$ is nonsingular. We can express $A + B$ as the product of two nonsingular matrices:

$$A + B = (I + BA^{-1})A.$$

Therefore $A + B$ is nonsingular, too, because its determinant will not be zero. Let us define the matrix $C \in \mathbb{R}^n$ such that

$$(A + B)^{-1} = A^{-1} + C.$$

Because $(A + B)(A + B)^{-1} = I$, it follows that

$$\begin{aligned}(A + B)(A^{-1} + C) = I \quad &\Leftrightarrow \quad I + AC + BA^{-1} + BC = I\\ &\Leftrightarrow \quad (A + B)C = -BA^{-1}\\ &\Leftrightarrow \quad C = -(A + B)^{-1}BA^{-1}\end{aligned}$$

So $(A + B)^{-1} = A^{-1} + C = A^{-1} - (A + B)^{-1}BA^{-1}$. If we take norms, we get

$$\|(A + B)^{-1}\| \leq \|A^{-1}\| + \|(A + B)^{-1}\|\|BA^{-1}\|.$$

Which is equivalent to

$$(I - \|BA^{-1}\|) \cdot \|(A + B)^{-1}\| \leq \|A^{-1}\|.$$

This gives the right inequality. The proof of the left inequality is similar. □

Rule of thumb.
If the condition number $\kappa(A) = 10^k$, then one may lose up to k digits of accuracy when solving a system with condition number k. This rule is determined by experience.

Remark.
The rule of thumb shows that we need to be careful with matrices that have huge condition numbers, because we can get solutions which are really far from the exact ones. When it comes to engineering for example, this sounds quite dangerous.

Definition 2.3 (Preconditioner).
A preconditioner P of a matrix $A \in \mathbb{R}^{n\times n}$ is a matrix such that

$$\kappa(P^{-1}A) \leq \kappa(A).$$

Remark.
Preconditioned techniques are often used in numerical methods, e.g. for solving linear systems of equations. In that case we solve $P^{-1}A\mathbf{x} = P\mathbf{b}$ instead of the original system, which reduces the computational costs, and is more effective. We will see an application for this when presenting the article [5] on page 10.

Definition 2.4 (Ill-conditioned and well-conditioned linear systems).
A system of linear equations is said to be ill-conditioned when some small perturbation in the system can produce relatively large changes in the exact solution. Otherwise, the system is said to be well-conditioned.

Remark.
If the condition number is much bigger than 1, the system is likely to be ill-conditioned. Sometimes we cannot calculate the determinant and the condition number of a matrix, because the arithmetical operations used here are similar to the ones used for solving the linear system of equations. That is why many times we do not care about the exact number, it is enough to know if the condition number is much larger than 1.

Remark.
The geometric meaning of an ill-conditioned system in two dimensions is trying to find the intersection point of two almost parallel lines.

Example.
Suppose that we have the results of an experiment, with the small uncertainty of 0.001 in the right hand side vector $\mathbf{b}$.

$$\begin{aligned} 0.835x + 0.667y &= b_1 \\ 0.333x + 0.266y &= b_2 \end{aligned}$$

With $(b_1, b_2) = (0.168, 0.067)$ the exact solution is $(x, y) = (1, -1)$. Let us make a small perturbation in $\mathbf{b}$ within the range of uncertainty of the measurements. If $(b_1, b_2) = (0.169, 0.066)$, the solution will change to $(x, y) = (-932, 1167)$. And if $(b_1, b_2) = (0.167, 0.068)$, the solution will change to $(x, y) = (934, -1169)$. We can see that a small change in the problem we try to solve, caused a big change in the solution. Taking the 1-norm, we get that the condition number of the coefficient matrix is 1,754,300, which is clearly not close to 1, therefore we can state, that the system is ill-conditioned. As we can see, the problem is that all three solutions have the same validity for being the exact one, and they are really far from each other, so we cannot rely on the solution we get from this system.

The main cause of ill-conditioned systems is that many times we cannot have exact values for the coefficient matrix, because the data is gained from experiments. This means that we cannot rely on the solution we get from this system, since the measurements used for calculating are not accurate. If they have a little error, the system gives a solution which is relatively really far from the other one. We cannot risk so much while building a bridge, or having an operation, for example.

2.2.1 Recent methods for solving ill-conditioned systems

It is obvious that ill-conditioned systems play an important role in several areas, hence there are a lot of scientists working on how to deal with these systems. I would like to present three different methods for this problem from recent articles. Two of them will be shown now, and one later (in subsection *4.3.2*), since it requires some further knowledge discussed in *4.3*.

1. ***Multi-scale method for ill-conditioned linear systems*** [12]

This article introduces a multi-scale method, which reduces the condition number of the coefficient matrix, thus, it gives an efficient way for solving linear ill-conditioned systems. After rewriting the given problem, due to the lowered condition number, it can be easily solved by a known iterative method. Then we only need to do some basic multiplications to find the solution of the original problem.

The method.

If the coefficient matrix A of the equations $A\mathbf{x} = \mathbf{b}$ (see (1)) is ill-conditioned, the system can be solved by the following equations

$$\tilde{A}\tilde{\mathbf{x}} = \mathbf{b}, \tag{2}$$

where $\tilde{A}$ and $\tilde{\mathbf{x}}$ are as follows

$$\tilde{A} = \begin{pmatrix} \frac{1}{\|\boldsymbol{\alpha_1}\|_2} a_{11} & \frac{1}{\|\boldsymbol{\alpha_2}\|_2} a_{12} & \dots & \frac{1}{\|\boldsymbol{\alpha_n}\|_2} a_{1n} \\ \frac{1}{\|\boldsymbol{\alpha_1}\|_2} a_{21} & \frac{1}{\|\boldsymbol{\alpha_2}\|_2} a_{22} & \dots & \frac{1}{\|\boldsymbol{\alpha_n}\|_2} a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{\|\boldsymbol{\alpha_1}\|_2} a_{n1} & \frac{1}{\|\boldsymbol{\alpha_2}\|_2} a_{n2} & \dots & \frac{1}{\|\boldsymbol{\alpha_n}\|_2} a_{nn} \end{pmatrix}, \quad \tilde{\mathbf{x}} = \begin{pmatrix} \|\boldsymbol{\alpha_1}\|_2 \ \mathbf{x_1} \\ \|\boldsymbol{\alpha_2}\ \|_2 \ \mathbf{x_2} \\ \vdots \\ \|\boldsymbol{\alpha_n}\|_2 \ \mathbf{x_n} \end{pmatrix},$$

here $\boldsymbol{\alpha_j}, \quad j = 1, 2, ...n \quad$ is the j^{th} column of A.

With (2), the following multi-scale method is obtained for system (1):
Step 1: Solve $\tilde{A}\tilde{\mathbf{x}} = \mathbf{b}$ with iterative methods

Step 2: $x = \begin{pmatrix} \frac{1}{\|\boldsymbol{\alpha_1}\|_2 \ \mathbf{x_1}} \\ \frac{1}{\|\boldsymbol{\alpha_2}\ \|_2 \ \mathbf{x_2}} \\ \vdots \\ \frac{1}{\|\boldsymbol{\alpha_n}\|_2 \ \mathbf{x_n}} \end{pmatrix}$.

2. ***A fast and efficient algorithm for solving ill-conditioned linear systems.*** [5]

This article presents an algorithm which is based on a preconditioned technique. The point is to find the LU factorization of A instead of computing A^{-1}, which reduces the computational costs, and we use the approximate inverse of L as a left preconditioner in the method. The algorithm first finds the usual LU factorization with partial pivoting, and applies the iterative refine method for the solution. In case it does not satisfy the stopping criteria, the preconditioned technique is applied, and the iterative refinement method is used on the approximate solution. This second part, as planned, only needs to be executed if the system is ill-conditioned, so it does not add any additional computational cost if the system is well-conditioned.

The algorithm.
We assume that LU approximates A ($A \approx LU$) with $\kappa(A) \approx \kappa(L)$. Let X_L denote an approximate inverse of L. By using X_L as a preconditioner, the linear system (1) can be transformed into

$$X_L A\mathbf{x} = X_L \mathbf{b}.$$

Remark. In the following steps, while presenting the algorithm, we will execute the LU factorization of A^T, but working with the LU factorization of A would also be correct.

Step 1. Execute an LU factorization of A^T with partial pivoting:

$$PA^\mathsf{T} \approx LU.$$

Then solve

$$U^\mathsf{T} L^\mathsf{T} P\mathbf{x} = \mathbf{b}$$

by forward and back substitutions for obtaining its approximate solution,

$$\tilde{\mathbf{x}} \approx P^\mathsf{T}(L^\mathsf{T})^{-1}(U^\mathsf{T})^{-1}\mathbf{b}.$$

Step 2. Apply the iterative refinement method to the approximate solution obtained at *Step 1.* By this method an accurate solution of $A\mathbf{x} = \mathbf{b}$ can be obtained if A is not ill-conditioned. So first, calculate the

$$\mathbf{r_k} \approx \mathbf{b} - A\tilde{\mathbf{x}}_\mathbf{k}$$

residual precisely. Next, we solve

$$A\mathbf{y} = \mathbf{r_k} \tag{3}$$

by using LU factors obtained at *Step 1.* Let $\tilde{\mathbf{y}}_\mathbf{k}$ be the approximate solution of (3), then we update $\tilde{\mathbf{x}}_\mathbf{k}$ by

$$\tilde{\mathbf{x}}_{\mathbf{k+1}} = \tilde{\mathbf{x}}_\mathbf{k} + \tilde{\mathbf{y}}_\mathbf{k}.$$

The more steps repeat, the more accuracy of $\tilde{\mathbf{x}}$ gains. If the stopping criterion for the iterations is satisfied, then the algorithm successfully stops. Otherwise, go to *Step 3.*

Step 3. Adopt a result of an LU factorization of A^T in *Step 1* such as

$$X_L \approx (U^\mathsf{T})^{-1}.$$

Use X_L as a left preconditioner of A to reduce the condition number of A. Multiply $A\mathbf{x} = \mathbf{b}$ by X_L from the left. Let C and $\mathbf{d}$ denote an approximation of $X_L A$ and $X_L\mathbf{b}$, respectively:

$$C \approx X_L A, \quad \mathbf{d} \approx X_L\mathbf{b}.$$

Here, $X_L A$ and $X_L\mathbf{b}$ should be calculated accurately. Then, we split both X_L and A into two parts, and

$$X_L A = ((X_L)_1 + (X_L)_2)\,(A_1 + A_2) = (X_L)_1 A_1 + (X_L)_1 A_2 + (X_L)_2 A,$$

which involves three triangular matrix-matrix multiplications. Now consider

$$C\mathbf{x} = \mathbf{d}. \tag{4}$$

We expect C is not ill-conditioned. If that is the case, (4) can be solved using a standard way with an LU factorization and forward and back substitutions.

Step 4. Apply the iterative refinement method to the approximate solution at *Step 3.* It is almost the same as *Step 2*, except

$$\mathbf{r_k} = X_L(\mathbf{b} - A\tilde{\mathbf{x}}_\mathbf{k}).$$

3 Solving nonlinear equations

For writing this section I used the books [3], [13].

Consider a continuous function f on a given interval $[a, b]$. Solving a nonlinear equation in one variable means finding $x \in [a, b]$, that satisfies $f(x) = 0$ (the root of f). If $f : \mathbb{R}^n \to \mathbb{R}^n$ is a multivariable function, then we search for the vector $\mathbf{x} \in \mathbb{R}^n$ for which $f(\mathbf{x}) = 0$ holds. In simpler cases the root can be found analytically, but we can not always use a formula to find the solution. Then we can apply numerical methods, which will give an approximate of the true solution. These methods stop when the required accuracy is reached, so when the value of f at the numerical and the true solution is closer than a predefined value. There are bracketing and open methods. The bracketing methods need a starting interval where the solution can be found. Then they keep the solution within the interval, while reducing the length of it small enough. The open methods start with an initial guess that is close to the actual solution. This may be done by looking at the graph of the function. Then by using a scheme the estimates are getting better. These latter ones are more effective, but does not always converge, whereas the bracketing methods do.

First we define three types of convergence types that appear in the following methods. [9]

Definition 3.1 (Linear convergence).
Suppose we have a sequence $\{x_n\}$ such that $x_n \to x_\infty \in \mathbb{R}^k$. We say that the convergence is linear if there exists $r \in (0, 1)$ such that

$$\frac{\|x_{n+1} - x_\infty\|}{\|x_{n+1} - x_\infty\|} \leq r$$

for all sufficiently large n.

Definition 3.2 (Superlinear convergence).
We say a sequence $\{x_n\}$ converges to x_∞ superlinearly if we have

$$\lim_{n\to\infty} \frac{\|x_{n+1} - x_\infty\|}{\|x_{n+1} - x_\infty\|} = 0.$$

Definition 3.3 (Quadratic convergence).
We say a sequence $\{x_n\}$ converges to x_∞ at a quadratic rate if $\exists\ 0 < M < \infty$ such that

$$\frac{\|x_{n+1} - x_\infty\|}{\|x_{n+1} - x_\infty\|^2} \leq M$$

for all sufficiently large n.

3.1 Bisection method

Bisection method is a bracketing method for finding a numerical solution for the equation $f(x) = 0$, when it is known that f is continuous within a given interval $[a, b]$ and the equation has a solution.

The algorithm.

Step 1. Choose the first interval by finding points a and b such that a solution exists between them. This means that $f(a)$ and $f(b)$ have different signs such that $f(a)f(b) <0$. The points can be determined by examining the plot of $f(x)$ versus x.

Step 2. Calculate the first estimate of the numerical solution x_{NS1} by

$$x_{NS1} = \frac{a+b}{2}.$$

Step 3. Determine whether the true solution is between a and x_{NS1}, or between x_{NS1} and b. This is done by checking the sign of the product $f(a)f(x_{NS1})$:
If $f(a)f(x_{NS1}) < 0$, the true solution is between a and x_{NS1}. $\Longrightarrow$
Let the new interval $[a, b] := [a, x_{NS1}]$.
If $f(a)f(x_{NS1}) > 0$, the true solution is between x_{NS1} and b. $\Longrightarrow$
Let the new interval $[a, b] := [x_{NS1}, b]$.

Step 2.-3. are repeated until the required accuracy is reached.

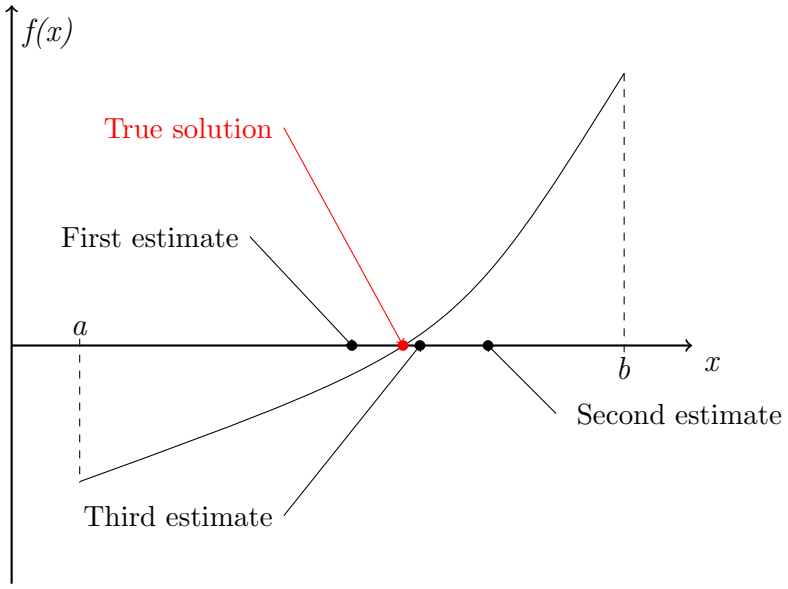


Figure 1: Bisection method.

Notes.

- The method converges linearly to the solution with convergence ratio 0.5 (provided a root was trapped in the interval $[a, b]$ to begin with).
- However, the method my fail when the function is tangent to the axis and does not cross the x-axis at $f(x) = 0$.

The code for Bisection method can be found in the Appendix *A.5*.

3.2 Regula falsi method

This method (also called false position method) is also a bracketing method for finding a numerical solution of an equation of the form $f(x) = 0$. We use it when it is known that, within a given interval $[a, b]$, $f(x)$ is continuous and the equation has a solution. The Regula falsi method is really similar to the Bisection method, the difference is how we choose the next estimate of the numerical solution.

The method starts with finding an initial interval $[a_1, b_1]$ that brackets the solution. After this we connect $f(a_1)$ and $f(b_1)$ with a straight line. The first estimate x_{NS1} of the numerical solution is the intersection point of this line and the x-axis. Then we choose the new interval $[a_2, b_2]$ such that it is a subinterval of $[a_1, b_1]$ that contains the solution. We continue this until our estimate is close enough to the true solution.

For a given interval $[a, b]$, the equation of a straight line that connects point $(b, f(b))$ to point $(a, f(a))$ is given by:

$$y = \frac{f(b) - f(a)}{b - a}(x - b) + f(b) \tag{5}$$

To find where this line intersects the x-axis, we need to substitute $y = 0$ to equation (5). Now if we solve the equation for x, we will get the numerical solution:

$$x_{NS} = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}. \tag{6}$$

The algorithm.

Step 1. Similarly to the Bisection method, first find points a and b such that $f(a)f(b) < 0$. We know that the solution is between a and b.

Step 2. Calculate the first estimate of the numerical solution x_{NS1} by using equation (6).

Step 3. Determine the next interval by checking the sign of the product $f(a) \cdot f(x_{NS1})$.

$$\text{If } f(a)f(x_{NS1}) < 0 \implies [a, b] := [a, x_{NS1}].$$
$$\text{If } f(a)f(x_{NS1}) > 0 \implies [a, b] := [x_{NS1}, b].$$

Repeat *Step 2.-3.* until a specified tolerance or error bound is attained.

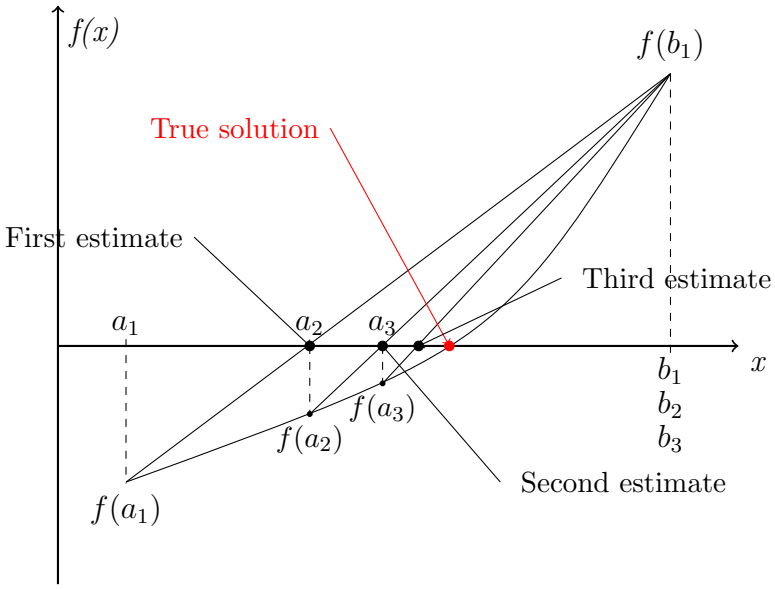


Figure 2: Regula falsi method.

Proposition 3.1.
Let f be three times continuously differentiable in a neighbourhood of the solution x_S with $f'(x_S) = 0$, $f"(x_S) > 0$. If the the root is close enough to the endpoints of the starting interval, then the method converges superlinearly to x_S with the order of convergence $\frac{1+\sqrt{5}}{2}$. [8]

3.3 Newton's method

Newton's method is a scheme for finding a numerical solution of an equation of the form $f(x) = 0$, where $f(x)$ is continuous and differentiable, and the equation is known to have a solution near a given point.

The algorithm starts by choosing the first estimate x_1 of the solution. We obtain the second estimate x_2 by taking the tangent line to $f(x)$ at the point $(x_1, f(x_1))$ and finding the intersection point of the tangent line with the x-axis. The next

estimate x_3 is the intersection of the tangent line to $f(x)$ at the point $(x_2, f(x_2))$ with the x-axis, and so on.

The slope $f'(x_1)$ for the first iteration is given by

$$f'(x_1) = \frac{f(x_1) - 0}{x_1 - x_2}.$$

If we rearrange the equation, we get

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

We can generalize this equation for determining the next estimate x_{i+1} form x_i:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{7}$$

This is the general iteration formula for Newton's method.

The algorithm.

Step 1. Choose a point x_1 as an initial guess of the solution.

Step 2. Calculate x_{i+1} for $i = 1, 2, ...$ by using equation (7), until the error is smaller than a specified value.

Proposition 3.2.

If f is three times continuously differentiable in a neighbourhood of the solution x_S with $f'(x_S) = 0$, $f"(x_S) > 0$, then Newton's method converge to x_S quadratically (provided that the starting point x_1 is close enough to x_S). [8]

Notes.

- When the method does not converge, it is usually because the starting point is not close enough to the solution. Convergence problems typically occur when the value of $f'(x)$ is close to zero in the vicinity of the solution.
- The derivative may be difficult to determine in some cases. This time we may determine the slope numerically, or we can use the Secant method, described in the next subsection, since it does not use the derivatives.

The code for Newton's method can be found in the Appendix *A.6*.

3.4 Secant method

The Secant method is a scheme for finding a numerical solution of the equation $f(x) = 0$. The method uses two points in the neighbourhood of the solution to find the new estimate. These two points are used to determine a straight line (secant line), and the point where it intersects the x-axis is the new estimate.

The slope of the secant line is given by

$$\frac{f(x_1) - f(x_2)}{x_1 - x_2} = \frac{f(x_2) - 0}{x_2 - x_3}.$$

This can be solved for x_3:

$$x_3 = x_2 - \frac{f(x_2)(x_1 - x_2)}{f(x_1) - f(x_2)}.$$

Once the point x_3 is determined, it is used together with the point x_2 to calculate the next estimate of the solution, x_4.

The generalized formula for x_{i+1} is

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)}. \tag{8}$$

Relationship to Newton's method.

Examination of the Secant method shows that when the two points that define the secant line are close to each other, the method is actually an approximated form of Newton's method. This can be seen by rewriting equation (8) in the form

$$x_{i+1} = x_i - \frac{f(x_i)}{\frac{f(x_{i-1}) - f(x_i)}{x_{i-1} - x_i}}.$$

The convergence rate of Secant method is $\frac{1+\sqrt{5}}{2}$, so it converges slower than Newton's method. However, the latter needs the actual derivative of f, whereas Secant method uses the approximation of it, saving computational costs.

The code for Secant method can be found in the Appendix *A.7*.

4 Eigenvalue problems

In this section I use the following books: [3], [4], [6], [7], [11].

Eigenvalue problems play a very important role in science and engineering. They appear whenever something is oscillating in a periodic motion. One famous example is related to the Chladni figures, which appear when fine grained sand or dust on a vibrating plate organizes itself to reveal the nodal lines of the vibrating plate. (Nodal lines are the lines that stay in place while the other parts are in the state of the vibration.) A spectacular video of Chldani figures can be seen on the link [1]. It shows how the geometric patterns become more and more complex as the pitch of the voice made increases. Also, the spectacular failure of the Tacoma bridge is shown to be related to an eigenvalue problem. To find further details about these examples, see [2].

In the study of vibrations, the eigenvalues represent the natural frequencies of a system or component, and the eigenvectors represent the modes of these vibrations. It is important to identify these natural frequencies, because when the system or component is subjected to periodic external loads (forces) at or near these frequencies, resonance can cause the response (motion) of the structure to be amplified, which potentially leads to failure of the component. In mechanics of materials , the principal stresses are the eigenvalues of the stress matrix, and the principal directions are the directions pf the associated eigenvectors. In quantum mechanics, the eigenvectors represent one of the states in which an object or a system may exist corresponding to a particular eigenvalue.

Although eigenvalue problems are often formulated at the level of differential equations, there is a link between these and eigenvalue problems involving $A\mathbf{u} = \lambda\mathbf{u}$. Numerical determination of the eigenvalues in a problem involving an ordinary differential equation reduces to finding the eigenvalues of an associated matrix A, which results in a problem of the form $A\mathbf{u} = \lambda\mathbf{u}$.

Definition 4.1 (Eigenvalues and Eigenvectors).
Given $A \in \mathbb{R}^{n\times n}$, find $\mathbf{x} \in \mathbb{R}^n, \mathbf{x} \neq 0$, and $\lambda \in \mathbb{C}$ such that $A\mathbf{x} = \lambda\mathbf{x}$. Here λ is an eigenvalue, $\mathbf{x}$ is an eigenvector of A. Any such pair $(\lambda, \mathbf{x})$ is called an eigenpair of A.

Remark (Geometrical interpretation).
Geometrically, $A\mathbf{x} = \lambda\mathbf{x}$ means that under transformation by A, the eigenvectors can only change in their magnitude or sign.

The orientation of the vectors $\mathbf{x}$ and $A\mathbf{x}$ is the same. The eigenvalue λ determines the amount of stretch or shrink made on the eigenvector when being transformed by A.

Definition 4.2 (Spectrum).
Let $A \in \mathbb{R}^{n\times n}$. The set of distinct eigenvalues of A, denoted by $\sigma(A)$, is called the spectrum of A.

Definition 4.3 (Spectral radius).
Let $A \in \mathbb{R}^{n\times n}$. The number $\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda|$ is called the spectral radius of A.

Many applications do not require precise knowledge of the eigenvalues of A, only an upper bound. The next proposition gives a rough but cheap approximate for $\rho(A)$.

Proposition 4.1.
$\forall A \in \mathbb{R}^{n\times n}: \quad \rho(A) \leq \|A\|$ for every matrix norm.

Proof. For any $(\lambda, \mathbf{x})$ eigenpair of A, $A\mathbf{x} = \lambda\mathbf{x}$. We know that for any vector norm $\|\lambda\mathbf{x}\| = |\lambda| \cdot \|\mathbf{x}\|$, therefore

$$|\lambda| \cdot \|\mathbf{x}\| = \|\lambda\mathbf{x}\| = \|A\mathbf{x}\| \leq \|A\| \cdot \|\mathbf{x}\|$$

for all $\lambda \in \sigma(A)$. □

By computing this eigenvalue bound we only get one big circle, whose radius is usually much bigger than the spectral radius $\rho(A)$. We get a better approximation with using Gershgorin circles, which are elaborated in the next subsection (*4.1*).

Earlier we analysed how the solution of the system
$A\mathbf{x} = \mathbf{b}, A \in \mathbb{R}^{n\times n},\ \mathbf{b} \in \mathbb{R}^n,\ \mathbf{x} \in \mathbb{R}^n$ changes if A or $\mathbf{b}$ has some uncertainties. Now we want to see how the eigenvalues of a matrix A change if the elements of A have small perturbations. The following theorem tells us about the condition of the eigenvalues of A in case the Jordan decomposition of A is diagonal.

Theorem 4.2 (Conditioning of eigenvalues).
Let $A \in \mathbb{C}^{n\times n}$ and $A = S^{-1}DS$, where $D = \text{diag}(\lambda_1, \lambda_2, ...\lambda_n)$. Let $\|A\|$ matrix norm be such that for all D diagonal matrix $\|D\| = \max_i |d_i|$. (E.g. 1-norm, 2-norm, ∞-norm.) If $B \in \mathbb{C}^{n\times n}$ and $\epsilon > 0$, then in case

$$\lambda(\epsilon) \in \sigma(A + \epsilon B),$$

$\exists\ i$ such that

$$|\lambda(\epsilon) - \lambda_i| \ \leq\ \epsilon\ \|B\|\ \|S\|\ \|S^{-1}\| \ =\ \epsilon\ \|B\|\ \kappa(S).$$

Proof.
Let $\lambda(\epsilon)$ be an eigenvalue of $A + \epsilon B$. If $\lambda(\epsilon) = \lambda_i$ for some i, then the left hand side of the inequality is 0, and the right hand side consists of only nonnegative numbers, so the statement is trivially true.

Therefore we may assume that $\lambda(\epsilon) \neq \lambda_i$ for $i = 1, ...n$. In this case

$$\begin{aligned} 0 &= \det(A + \epsilon B - \lambda(\epsilon)I) = \det(S^{-1}DS + \epsilon B - \lambda(\epsilon)I) = \\ &= \det S^{-1}(D + \epsilon SBS^{-1} - \lambda(\epsilon)I)S) = \det(D - \lambda(\epsilon)I + \epsilon SBS^{-1}) = \\ &= \det(D - \lambda(\epsilon)I)\det(I + \epsilon(D - \lambda(\epsilon)I)^{-1}SBS^{-1}) \end{aligned}$$

(Rather than doing small manipulations on the equations, we used the property that the determinant of similar matrices are equal.)

Since $\lambda(\epsilon) \neq \lambda_i$ for $i = 1, ...n$, we have $\det(D - \lambda(\epsilon)I) \neq 0$. For this reason

$$\det(I + \epsilon(D - \lambda(\epsilon)I)^{-1}SBS^{-1}) = 0.$$

This implies that

$$\epsilon \, \|(D - \lambda(\epsilon)I)^{-1}\| \, \|S\| \, \|B\| \, \|S^{-1}\| \geq \|\epsilon(D - \lambda(\epsilon)I)^{-1}SBS^{-1}\| \geq 1,$$

because otherwise the matrix would be invertible, and its determinant would not be 0. Hence, $\epsilon \, \|B\| \, \kappa(S) \geq \min_i |\lambda_i - \lambda(\epsilon)|$. □

4.1 Gershgorin circles

In many cases it is not essential to know the eigenvalues exactly, but is enough to have a good approximation for them. Now we will see a graphic way of approximating eigenvalues.

Here consider an $n \times n$ complex matrix A (as the previous definitions are also true over $\mathbb{C}$).

Theorem 4.3 (Gershgorin I).
Let $A \in \mathbb{C}^{n \times n}$. For every row of the matrix, let R_i be the sum of the absolute values of the nondiagonal entries, so

$$R_i := \sum_{j=1, j \neq i}^{n} |a_{ij|}.$$

Let B_i be the discs in the complex plane with center a_{ii} and radius R_i, so

$$B_i := \{z \in \mathbb{C}, \quad |z - a_{ii}| \leq R_i\}.$$

Proof. Let λ be an eigenvalue of A and $\mathbf{u}$ be the corresponding eigenvector.

Then the i^{th} equation of the eigenvalue equation $A\mathbf{u} = \lambda\mathbf{u}$ gives

$$\sum_{j=1}^{n} a_{ij}\mathbf{u_j} = \lambda\mathbf{u_i} \qquad (i = 1, 2, ..., n).$$

Now choose m such that $\|\mathbf{u}\|_\infty = \max\limits_{l=1,2,...,n} |\mathbf{u_l}| = |\mathbf{u_m}|$. We obtain

$$\sum_{j=1, j\neq m}^{n} a_{mj}\mathbf{u_j} = (\lambda - a_{mm})\ \mathbf{u_m}. \tag{9}$$

Since $\mathbf{u}$ is an eigenvector, we know that $\mathbf{u_m} \neq 0$, therefore we can divide (9) by $\mathbf{u_m}$. Now taking the absolute value of both sides of the equality we get

$$|\lambda - a_{mm}| = \sum_{j=1, j\neq m}^{n} |a_{mj}| \cdot \left|\frac{\mathbf{u_j}}{\mathbf{u_m}}\right|.$$

Here $\left|\frac{\mathbf{u_j}}{\mathbf{u_m}}\right| \leq 1$, because $\mathbf{u_m} = \|\mathbf{u}\|$, which means that for all $j = 1..n \quad \mathbf{u_j} \leq \mathbf{u_m}$. Hence, if we leave out this factor from the right side of the equation, we will increase it. So, the conclusion is

$$|\lambda - a_{mm}| \leq \sum_{j=1, j\neq m}^{n} |a_{mj}| = R_m,$$

which means that $\lambda \in B_m$. □

Corollary.
The eigenvalues of a diagonal matrix are the entries on the diagonal.

Theorem 4.4 (Gershgorin II).
Let $A \in \mathbb{C}^{n\times n}$. Assume we can divide the Gershgorin discs into two disjoint sets: V_1 is the union of k discs and V_2 is the union of the other $n - k$ discs. Then k eigenvalues lie in V_1 and $n - k$ eigenvalues lie in V_2 (counting multiplicities).

Proof. Let $\epsilon \in [0, 1]$ and define the following matrix:

$$(B(\epsilon))_{ij} = \begin{cases} a_{ii}, & (i = j) \\ \epsilon a_{ij}, & (i \neq j). \end{cases}$$

One can easily see that in the case when $\epsilon = 0$, $B(0) = \mathrm{diag}(a_{11}, ..., a_{nn})$, the theorem is clearly true. Furthermore, using the definition, we can realize that for all $\epsilon \in [0, 1]$ the diagonal entries of the matrix $B(\epsilon)$ are the same as of the original matrix A. For this reason the centers of the Gershgorin circles will remain the same for any $\epsilon \in [0, 1]$ parameter.
The union of the k circles will be disjoint from the $n - k$ circles for all $B(\epsilon)$, $\epsilon \in [0, 1]$ matrices. Since V_1 and V_2 consist of closed discs and they are disjoint, their distance d is strictly larger than 0. We saw that the centers of the Gershgorin

circles are independent from the parameter ϵ, but the radii grow with ϵ growing bigger. The radii will be the largest for $\epsilon = 1$, in the case when $B(1) = A$. We know that even in this case the $d > 0$ distance applies. Now we obtained what we need to see that the theorem is true, because the eigenvalues are continuous functions of ϵ. □

Remark.
If A is real, the centers of the Gershgorin discs lie on the real axis.
For example, see Figure *4* and *5*.

Remark.
We know that for any real symmetric matrix the eigenvalues are real, which gives us a more exact approximation for the eigenvalues, namely the range of eigenvalues is going to be given by the intersection of the real axis and the union of the Gershgorin circles.

Example.
Consider the following matrix $A = \begin{pmatrix} -2 & 4 & -3 & 1 \\ 4 & 14 & 2 & 0 \\ -3 & 2 & -8 & -1 \\ 1 & 0 & -1 & 1 \end{pmatrix}$.

Figure *3* shows the Gershgorin discs of A and its eigenvalues denoted by red stars.

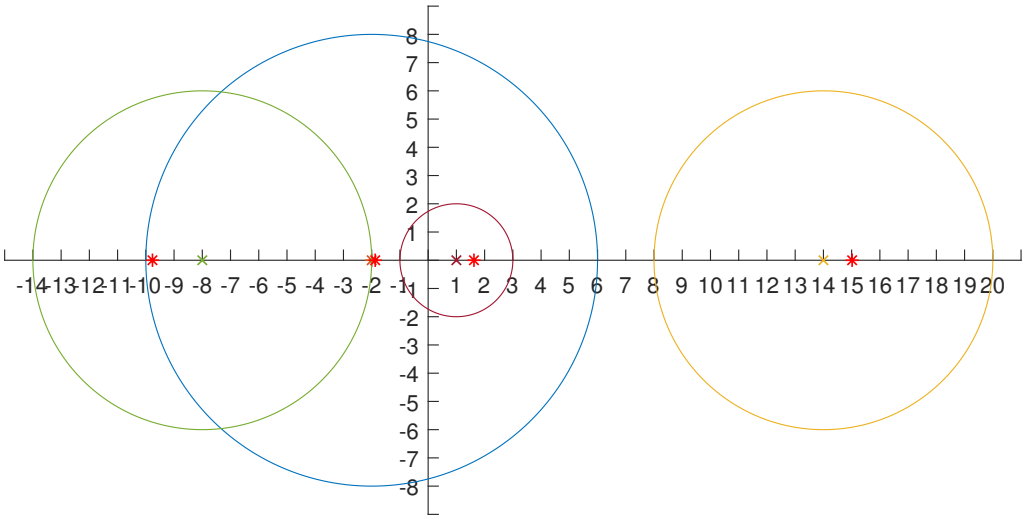


Figure 3: Gershgorin circles of a symmetric matrix

The following theorem shows a way for modifying the radii of Gershgorin circles with some parameters. This enables us to move or disjoint the circles when it is needed.

Theorem 4.5 (Gershgorin III).
Let $D = \text{diag}(d_1, d_2, ..., d_n) \in \mathbb{R}^{n\times n},\ \det(D) \neq 0$. Consider the matrix $\bar{A} = D^{-1}AD$. We know that it has the same eigenvalues as A (because similarity transformation preserves eigenvalues). Furthermore, we can see that diagonal entries do not change, which means, the centers of the Gershgorin circles remain the same, though the $\bar{r}_i$ radii of $\bar{A}$ will be modified in the following way:

$$\bar{r}_i = \sum_{j=1, j\neq i}^{n} \left| \frac{a_{ij}d_j}{d_i} \right| \qquad (i = 1, 2, ..., n).$$

.

Remark.
Since $\sigma(A^{\mathsf{T}}) = \sigma(A)$, we can also define the radii of Gershgorin circles with the column sums, so the eigenvalues of A are also contained in the following Gershgorin circles:

$$D_j := \{z \in \mathbb{C}, \quad |z - a_{ii}| \leq C_j\}$$

where

$$C_j = \sum_{i=1, i\neq j}^{n} |a_{ij}| \qquad (j = 1, 2, ..., n).$$

Remark.
Let us use the following notations: $\mathcal{G}_r = \bigcup_{i=1}^{n} B_i$ and $\mathcal{G}_c = \bigcup_{j=1}^{n} D_j$.
We can obtain the best estimate by considering $G_r \bigcap G_c$, which will be seen in the next example.

Example.
We would like to give an estimate for the eigenvalues of $A = \begin{pmatrix} 5 & 1 & 1 \\ 0 & 6 & 1 \\ 1 & 0 & -5 \end{pmatrix}$.
The exact solution is $\sigma(A) = \{5, (1 \pm 5\sqrt{5})/2\} \approx \{5, 6.0902, -5.0902\}$. Now we compare results from three methods.

- We can have a crude estimate from Proposition *4.1*, which gives us (using the ∞-norm) $|\lambda| \ \leq \ \|A\|_\infty = 7 \qquad \forall\lambda \in \sigma(A)$.
- Now we use the Gershgorin circles derived from row sum to give a better estimate. The computed discs can bee seen on the Figure *4*. Theorems *4.3* and *4.4* guarantee that one eigenvalue is in (or on) the circle centered at -5, while the remaining two eigenvalues are in (or on) the larger circle centered at +5. The red stars show the exact location of the eigenvalues on the plot.

- The best estimation is obtained from the intersection of the Gershgorin circles derived from $G_r \bigcap G_c$. Figure *5* shows us that one eigenvalue is in the circle centered at -5, and the other two eigenvalues are in the union of the other two circles. Again the red stars mark the places of the actual eigenvalues.

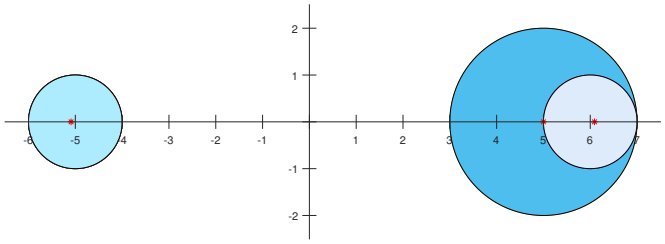


Figure 4: Gershgorin circles derived from row sum

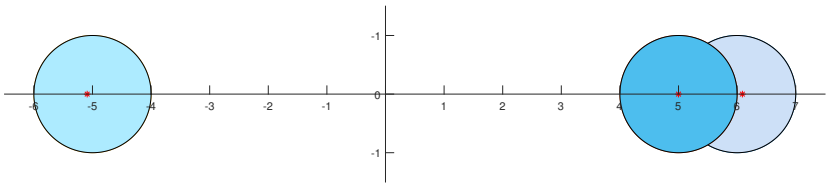


Figure 5: Gershgorin circles derived from $G_r \bigcap G_c$

4.2 Iteration methods

Since eigenvalues are the roots of the characteristic polynomial of a matrix, they can only be computed iteratively, as there are no closed formulas for roots of polynomilas of degree higher than four.

4.2.1 Power iteration

This method computes one eigenpair.
Let $A \in \mathbb{R}^{n \times n}$ such that for the eigenvalues $\lambda_1, \lambda_2, ...\lambda_n$ of A the following holds: $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$. Here λ_1 is called the dominant eigenvalue. Let $\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_n}$ be the corresponding eigenvectors, and suppose that they are independent. Therefore they form a basis in $\mathbb{R}^n$.

Choose an arbitrary start vector $\mathbf{x_0} \in \mathbb{R}^n,\ \mathbf{x_0} \neq 0$. We can write $\mathbf{x_0}$ in the terms of the basis vectors $\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_n}$: $\mathbf{x_0} = \sum_{j=1}^{n} \beta_j \mathbf{u_j}$. Suppose that $\beta_1 \neq 0$, which means that $\mathbf{x_0}$ is not perpendicular to $\mathbf{u_1}$.
Consider the following iteration.

$$\mathbf{x_{i+1}} = A\mathbf{x_i},\ \text{for } i = 0, 1, 2, ...$$

Where do these vectors $\mathbf{x_0}, \mathbf{x_1}, \mathbf{x_2}, ...$ converge? The next proposition gives us the answer.

Proposition 4.6.
The vectors $\mathbf{x_i}$ $(i \to \infty)$ converge to $\mathbf{u_1}$, the eigenvector that belongs to the dominant eigenvalue λ_1.

Proof.

$$\mathbf{x_i} = A^i\mathbf{x_0} = A^i\left(\sum_{j=1}^{n}\beta_j\mathbf{u_j}\right) = \sum_{j=1}^{n}\beta_j A^i\mathbf{u_j} = \sum_{j=1}^{n}\beta_j\lambda_j^i\mathbf{u_j} = \beta_1\lambda_1^i\mathbf{u_1} + \sum_{j=2}^{n}\beta_j\lambda_j^i\mathbf{u_j}$$
$$= \lambda_1^i\left(\beta_1\mathbf{u_1} + \sum_{j=2}^{n}\beta_j\left(\frac{\lambda_j}{\lambda_1}\right)^i\mathbf{u_j}\right)$$

Since $|\lambda_1| > |\lambda_2| \geq ... \geq |\lambda_n|$, it follows that $\left(\frac{\lambda_j}{\lambda_1}\right)^i \to 0$ for $i \to \infty$.
So we obtained that $\mathbf{x_i} \to \lambda_1^i\beta_1\mathbf{u_1}$. □

There is always the question if we can speed up the convergence of the method. The following variant of power method shows a way for this.

Power method with shift.

Let us see the idea first. Let $\sigma \in \mathbb{R}$. If $A\mathbf{u} = \lambda\mathbf{u}$, then

$$(A - \sigma I)\mathbf{u} = A\mathbf{u} - \sigma\mathbf{u} = \lambda\mathbf{u} - \sigma\mathbf{u} = (\lambda - \sigma)\mathbf{u}.$$

This means that $B = A - \sigma I$ has the same eigenvectors as A, but the eigenvalues are shifted, so B has the eigenvalues $\mu_k = \lambda_k - \sigma$ with k=1,2,...,n.
Now use the power method on B. This is the power method with shift.
Of course the next question is how to choose σ so that we can speed up the convergence.
Recall: the eigenvalues are $\lambda_1 > \lambda_2 \geq ... \geq \lambda_n$.
If we want $\mu_1 = \lambda_1 - \sigma$ to remain the dominant eigenvalue, we need $\sigma < \frac{\lambda_1+\lambda_n}{2}$.
If we want $\mu_2 = \lambda_2 - \sigma$ to remain the next largest, we need $\sigma \leq \frac{\lambda_2+\lambda_n}{2}$.
The converge speed for the power method with shift is $\frac{\mu_2}{\mu_1} = \frac{\lambda_2-\sigma}{\lambda_1-\sigma}$.

It is going to be minimal if we choose $\sigma = \frac{\lambda_2+\lambda_n}{2}$.

Remarks.
The optimal choice is theoretical, since λ_2 and λ_n are unknowns.
One can take a different shift in every iteration step.

4.2.2 Rayleigh quotient

Using the power method we get $\mathbf{x_{i+1}} \approx \lambda_1^{i+1}\beta_1\mathbf{u_1} = \lambda_1\lambda_1^i\beta_1\mathbf{u_1} \approx \lambda_1\mathbf{x_i}$. If we take the inner product with $\mathbf{x_i}$ on both sides, we have the following approximation for the dominant eigenvalue: $\lambda_1 \approx \frac{<\mathbf{x_{i+1}},\mathbf{x_i}>}{<\mathbf{x_i},\mathbf{x_i}>}$

Definition 4.4 (Rayleigh quotient).
Let $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq 0$. The Rayleigh quotient $\rho(\mathbf{x}) \in \mathbb{R}$ of $\mathbf{x}$ is defined as

$$\rho(\mathbf{x}) := \frac{< A\mathbf{x}, \mathbf{x} >}{< \mathbf{x}, \mathbf{x} >}.$$

We can see that the Rayleigh quotients $\rho(\mathbf{x_i})$ converge to the dominant eigenvalue λ_1.

Theorem 4.7.
Assume that the eigenvectors $\mathbf{u_i}$ are normalized, so $\|\mathbf{u_i}\|_2 = 1$.
Let $\mathbf{x} \in \mathbb{R}^n$ with $\|\mathbf{x}\|_2 = 1$ be an approximation of $\mathbf{u_1}$.
Then the Rayleigh quotients $\rho(\mathbf{x}) = \frac{<A\mathbf{x},\mathbf{x}>}{<\mathbf{x},\mathbf{x}>} = < A\mathbf{x}, \mathbf{x} >$ is an approximation of λ_1 and the estimated error is

$$|\lambda_1 - \rho(\mathbf{x})| \ \leq\ 2 \cdot \|A\|_2 \cdot \|\mathbf{u_1} - \mathbf{x}\|_2.$$

Proof.

$$\begin{aligned}
|\lambda_1 - \rho(\mathbf{x})| &= |\lambda_1 < \mathbf{u_1}, \mathbf{u_1} > - < A\mathbf{x}, \mathbf{x} > | \\
&= | < A\mathbf{u_1}, \mathbf{u_1} > - < A\mathbf{x}, \mathbf{x} > | \\
&= | < A\mathbf{u_1}, \mathbf{u_1} > - < A\mathbf{x}, \mathbf{u_1} > + < A\mathbf{x}, \mathbf{u_1} > - < A\mathbf{x}, \mathbf{x} > | \\
&= | < A(\mathbf{u_1} - \mathbf{x}), \mathbf{u_1} > + < A\mathbf{x}, \mathbf{x} > | \\
&\leq | < A(\mathbf{u_1} - \mathbf{x}), \mathbf{u_1} > | + | < A\mathbf{x}, \mathbf{x} > | \\
&\leq \|A(\mathbf{u_1} - \mathbf{x})\|_2 \cdot \|\mathbf{u_1}\|_2 + \|A\mathbf{x}\|_2 \cdot \|\mathbf{u_1} - \mathbf{x}\|_2 \\
&\leq \|A\|_2 \cdot \|\mathbf{u_1} - \mathbf{x}\|_2 + \|A\|_2 \cdot \|\mathbf{x}\|_2 \cdot \|\mathbf{u_1} - \mathbf{x}\|_2 \\
&= 2 \cdot \|A\|_2 \cdot \|\mathbf{u_1} - \mathbf{x}\|_2
\end{aligned}$$

□

During the power method iterations the values $\|\mathbf{x_i}\|_2$ can become very large or small. Let us normalize the vectors, so take $\frac{\mathbf{x_i}}{\|\mathbf{x_i}\|_2}$. This brings us to the following iteration.

Rayleigh quotient iteration.

Choose a start vector $\mathbf{z_0} \in \mathbb{R}^n$ with $\|\mathbf{z_0}\|_2 = 1$. For i =0,1,2,... compute

1. $\mathbf{y_{i+1}} := A\mathbf{z_i}$
2. $\rho_i := \rho(\mathbf{z_i}) = < A\mathbf{z_i}, \mathbf{z_i} > = < \mathbf{y_{i+1}}, \mathbf{z_i} >$
3. $\mathbf{z_{i+1}} := \frac{\mathbf{y_{i+1}}}{\|\mathbf{y_{i+1}}\|_2}$

The Rayleigh quotients ρ_i converge to the dominant eigenvalue λ_1.

Connection between Power method and Rayleigh quotient iteration.

Power method	**Rayleigh quotient iteration**
Choose $\mathbf{x_0} \in \mathbb{R}^n$	Choose $\mathbf{z_0} \in \mathbb{R}^n$ with $\|\mathbf{z_0}\|_2 = 1$
Compute for i =0,1,2,...	Compute for i =0,1,2,...
1. $\mathbf{x_{i+1}} := A\mathbf{x_i}$	1. $\mathbf{y_{i+1}} := A\mathbf{z_i}$
2. $\rho(\mathbf{x_i}) = \frac{<\mathbf{x_{i+1}}, \mathbf{x_i}>}{<\mathbf{x_i}, \mathbf{x_i}>}$	2. $\rho_i := \rho(\mathbf{z_i}) = < A\mathbf{z_i}, \mathbf{z_i} > = < \mathbf{y_{i+1}}, \mathbf{z_i} >$
	3. $\mathbf{z_{i+1}} := \frac{\mathbf{y_{i+1}}}{\|\mathbf{y_{i+1}}\|_2}$

Assume that we take $\mathbf{x_0} = \mathbf{z_0}$. What is the relationship between $\mathbf{x_i}$ and $\mathbf{z_i}$?

Proposition 4.8.

Using the previous notations, we have

$$\mathbf{z_i} = \frac{\mathbf{x_i}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_i}\|_2} \quad \text{and} \quad \rho(\mathbf{z_i}) = \rho(\mathbf{x_i}).$$

Proof.

We will prove $\mathbf{z_i} = \frac{\mathbf{x_i}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_i}\|_2}$ by induction.

Base case (k=1):

$$\mathbf{z_1} = \frac{\mathbf{y_1}}{\|\mathbf{y_1}\|_2} = \frac{A\mathbf{z_0}}{\|\mathbf{y_1}\|_2} = \frac{A\mathbf{x_0}}{\|\mathbf{y_1}\|_2} = \frac{\mathbf{x_1}}{\|\mathbf{y_1}\|_2}.$$

Induction hypothesis (IH): We assume that the claim holds for any $k \geq 1$, which means that $\mathbf{z_k} = \frac{\mathbf{x_k}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_k}\|_2}$ for $k \geq$1. We prove the claim for k+1.

$$\mathbf{z_{k+1}} = \frac{\mathbf{y_{k+1}}}{\|\mathbf{y_{k+1}}\|_2} = \frac{A\mathbf{z_k}}{\|\mathbf{y_{k+1}}\|_2} = \frac{A(\frac{\mathbf{x_k}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_k}\|_2})}{\|\mathbf{y_{k+1}}\|_2} \qquad \text{(by the IH)}$$

$$= \frac{A\mathbf{x_k}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_k}\|_2 \cdot \|\mathbf{y_{k+1}}\|_2} = \frac{\mathbf{x_{k+1}}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_{k+1}}\|_2}$$

The proof of the second part comes now.

$$\begin{aligned}
\rho(\mathbf{z_i}) &= < A\mathbf{z_i}, \mathbf{z_i} > = < \mathbf{y_{i+1}}, \mathbf{z_i} > = \ \|\mathbf{y_{i+1}}\|_2 < \mathbf{z_{i+1}}, \mathbf{z_i} > \\
&= \|\mathbf{y_{i+1}}\|_2 \frac{< \mathbf{z_{i+1}}, \mathbf{z_i} >}{< \mathbf{z_i}, \mathbf{z_i} >} \\
&= \|\mathbf{y_{i+1}}\|_2 \frac{\left\langle \frac{\mathbf{x_{i+1}}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_{i+1}}\|_2}, \frac{\mathbf{x_i}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_i}\|_2} \right\rangle}{\left\langle \frac{\mathbf{x_i}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_i}\|_2}, \frac{\mathbf{x_i}}{\|\mathbf{y_1}\|_2 \cdot \|\mathbf{y_2}\|_2 \cdot \ldots \cdot \|\mathbf{y_i}\|_2} \right\rangle} \\
&= \|\mathbf{y_{i+1}}\|_2 \left(\frac{< \frac{\mathbf{x_{i+1}}}{\|\mathbf{y_{i+1}}\|_2}, \mathbf{x_i} >}{< \mathbf{x_i}, \mathbf{x_i} >} \right) = \frac{< \mathbf{x_{i+1}}, \mathbf{x_i} >}{< \mathbf{x_i}, \mathbf{x_i} >} = \frac{< A\mathbf{x_i}, \mathbf{x_i} >}{< \mathbf{x_i}, \mathbf{x_i} >} = \rho(\mathbf{x_i})
\end{aligned}$$

□

This means that the Rayleigh quotients $\rho(\mathbf{z_i})$ in the scaled version of the power method have the same convergence speed as the quotients $\rho(\mathbf{x_i})$.

Convergence speed of the Rayleigh quotient iteration.

We have $\rho(\mathbf{x_i}) = \lambda_1 \left(1 + O\left(\left(\frac{\lambda_2}{\lambda_1}\right)^i\right)\right)$

This means that $\rho(\mathbf{x_i}) = \lambda_1 + c \cdot \left(\frac{\lambda_2}{\lambda_1}\right)^i$ and $\lim\limits_{i \to \infty} \frac{\rho(\mathbf{x_{i+1}}) - \lambda_1}{\rho(\mathbf{x_i}) - \lambda_1} = \frac{\lambda_2}{\lambda_1}$.

$$\implies \quad \underbrace{\rho(\mathbf{x_{i+1}}) - \lambda_1}_{\text{error in step i+1}} \approx \underbrace{\frac{\lambda_2}{\lambda_1}}_{\text{convergence factor}} \cdot \underbrace{(\rho(\mathbf{x_i}) - \lambda_1)}_{\text{error in step i}}$$

Remark.

The Rayleigh quotients $\rho(\mathbf{x_i})$ converge linearly to λ_1.

The convergence factor is $\frac{\lambda_2}{\lambda_1}$. So if $\left|\frac{\lambda_2}{\lambda_1}\right|$ is close to 1, the convergence is slow, and if it is $<< 1$, the method converges fast.

Special case of symmetric matrices.

If A is symmetric, then the the convergence speed of the power method is higher:

$$\rho(\mathbf{x_i}) = \lambda_1^{2i} \cdot \left(1 + O\left(\frac{\lambda_2}{\lambda_1}\right)^{2i}\right)$$

Here the convergence factor is $\left(\frac{\lambda_2}{\lambda_1}\right)^{2i}$.

4.2.3 Inverse iteration

Let A be nonsingular. Let λ be an eigenvalue with the corresponding eigenvector $\mathbf{u}$. Since A is nonsingular, we know that $\lambda \neq 0$. Then

$$A\mathbf{u} = \lambda\mathbf{u} \Leftrightarrow A^{-1}A\mathbf{u} = \lambda A^{-1}\mathbf{u} \Leftrightarrow \mathbf{u} = \lambda A^{-1}\mathbf{u} \Leftrightarrow A^{-1}\mathbf{u} = \frac{1}{\lambda}\mathbf{u}.$$

Therefore the matrix $B = A^{-1}$ has the same eigenvectors as A, but the eigenvalues μ_k of B are $\mu_k = \frac{1}{\lambda_k}$.
The inverse iteration is applying the power method on $B = A^{-1}$. It will find the dominant eigenvalue $\mu_n = \frac{1}{\lambda_n}$ with the convergence factor $\frac{\mu_{n-1}}{\mu_n} = \frac{\lambda_n}{\lambda_{n-1}}$.

Inverse iteration with shift.
We apply the power method on $B = (A - \sigma I)^{-1}$.
B has the same eigenvectors as A, with the shifted eigenvalues $\mu_k = \frac{1}{\lambda_k - \sigma}$.
What is the dominant eigenvalue? The good news are that we can make any eigenvalue dominant, because it depends on the choice of σ. This means that with the appropriate choice of σ we can compute any of the eigenvalues of A. When we have the result μ_k after the iteration, we just add σ to its reciprocal.
If we choose σ such that the dominant eigenvalue is μ_k, and the next largest is μ_l, then the convergence factor is $\frac{\mu_l}{\mu_k} = \frac{\lambda_k - \sigma}{\lambda_l - \sigma}$. To speed up the convergence, we can modify σ in every step. A good choice is to take the currently computed λ_k as the next shift.

4.2.4 Orthogonal iteration

The orthogonal or subspace iteration is a generalized version of the power method to compute more than one eigenpair. The idea is to use the power method on more vectors at the same time. So instead of a single vector $\mathbf{x} \in \mathbb{R}^n$, choose a start matrix $Q_0 \in \mathbb{R}^{n \times k}$. Make sure that the columns of Q_0 are orthonormal. The number of its columns, $1 < k \leq n$, is the number of the eigenvalues we want to compute.

Orthogonal iteration
1. Choose a start matrix Q_0 with orthonormal columns.
2. For i =0,1,2,... compute matrices Q_{i+1} and R_{i+1} such that

$$Q_{i+1} R_{i+1} = A Q_i,$$

where $Q_{i+1} \in \mathbb{R}^{n \times k}$ is orthogonal and $R_{i+1} \in \mathbb{R}^{k \times k}$ is upper triangular.

Remark.
Because R_i is upper triangular, the diagonal entries of R_i ($i \to \infty$) converge to the eigenvalues $\lambda_1, \lambda_2, ..., \lambda_k$.

Disadvantages of the orthogonal iteration.

- If A is a dense matrix, it is expensive to compute a QR decomposition in every step of the iteration.

- The start matrix Q_0 must be carefully chosen. Columns should have nonzero components in the directions of the eigenvectors we want to compute.
- The convergence speed is essentially that of the power method without shift.

4.2.5 QR iteration

Theorem 4.9 (QR decomposition.).
Consider the matrix $A \in {}^{n\times n}$ and assume that A^{-1} exists.
Then $\exists!$ $A = QR$ decomposition, where Q is orthogonal and R is upper triangular with $r_{ii} > 0$ for i =1,2,...,n.

Proof.
Now we will show only the uniqueness, assuming that there exists the decomposition $A = QR$. (We will prove the existence by giving an algorithm at the end of this chapter.)
Indirectly, assume that the QR decomposition exists, and it is not unique. It follows that $A = Q_1R_1 = Q_2R_2$. Multiplying the equation $Q_1R_1 = Q_2R_2$ by R_1^{-1} from the right and by $Q_2^{-1} = Q_2^{\mathsf{T}}$ from the left, we get $Q_2^{\mathsf{T}}Q_1 = R_2R_1^{-1}$. On the left side of the equation we have an orthogonal matrix, and on the right an upper triangular. An upper triangular matrix which is orthogonal (and therefore normal as well), is diagonal. A diagonal matrix with positive entries in its diagonal is the identity matrix. So the equality holds only if $Q_1 = Q_2$, and it implies that $R_1 = R_2$. □

The QR algorithm.

The QR algorithm was listed as one of the top ten algorithms in the last century. This is the most widely used algorithm for computing the eigenvalues of dense matrices. It finds all the eigenvalues and eigenvectors of the matrix.

First, we will transform the given matrix $A \in \mathbb{R}^{n\times n}$ into upper Hessenberg form. This will reduce the computing costs for finding the QR decomposition. The upper Hessenberg form is an upper triangular matrix with one subdiagonal. We will do the transformation by Givens rotations.

Definition 4.5 (Givens rotation).
A Givens rotation is a matrix of the form

$$
\mathrm{G}= \begin{array}{c} \\ \\ \\ \\ p \\ \\ \\ \\ q \\ \\ \\ \end{array}
\begin{array}{c}
\begin{array}{cccccccccc} p & & & & q & & & & & \end{array} \\
\begin{pmatrix}
1 & & & & & & & & & \\
& \ddots & & & & & & & & \\
& & 1 & & & & & & & \\
& & & c & & & s & & & \\
& & & & 1 & & & & & \\
& & & & & \ddots & & & & \\
& & & & & & 1 & & & \\
& & & -s & & & & c & & \\
& & & & & & & & 1 & \\
& & & & & & & & & \ddots \\
& & & & & & & & & & 1
\end{pmatrix}
\end{array}
$$

with $c^2 + s^2 = 1$.

Interpretation as mapping.
G is a rotation over the (p, q)-plane with the angle Φ where $c = \cos(\Phi)$ and $s = \sin(\Phi)$.

Properties of G.

- G is the identity matrix except the rows and columns p, q.

$$G_{pp} = c \qquad G_{pq} = s$$
$$G_{qp} = -s \qquad G_{qq} = c$$

- G is orthogonal, since $G^\mathsf{T} G = I$.
- We can create one zero entry in the matrix A with one Givens rotation.

So we will transform A into upper Hessenberg form by Givens rotations.
For j=1,2,...n-2 create zeros in the j^{th} column at positions $a_{j+2,j},\ a_{j+3,j}, ..., a_{n,j}$. To create a zero at position $(3, 1)$, we change the two entries $a_{2,1}$ and $a_{3,1}$. So we use a Givens rotation with p=2 and q=3. Let us denote this by $G_{2,3}$. If we left multiply A by $G_{2,3}$, only the rows 2 and 3 are changed. Similarly, if we right multiply A by $G_{2,3}^\mathsf{T}$, only the columns 2 and 3 are changed.
One can see that $G_{2,3} A G_{2,3}^\mathsf{T}$ has zero at position $(3, 1)$.
Define $\overline{A} = G_{n-1,n} G_{n-2,n} G_{n-2,n-1} ... G_{2,3} A G_{2,3}^\mathsf{T} ... G_{n-2,n-1}^\mathsf{T} G_{n-2,n}^\mathsf{T} G_{n-1,n}^\mathsf{T}$. Then $\overline{A}$ has upper Hessenberg form and we used a similarity transformation, so $\overline{A}$ has the same eigenvalues as A.

Definition 4.6 (Irreducibility).
A is reducible if there is a permutation matrix P such that PAP^T is block upper triangular.

So $PAP^\mathsf{T} = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix}$. Otherwise A is called irreducible.

Proposition 4.10.
If the Hessenberg matrix $\overline{A}$ is irreducible, then $Q_0 = I$ is a suitable start matrix for the orthogonal iteration on A.

Assume that $\overline{A}$ is irreducible.

Orthogonal iteration on the Hessenberg matrix.

1. Take as a start matrix $Q_0 = I$.
2. For i=0,1,2,... compute matrices Q_{i+1} and R_{i+1} such that $Q_{i+1}R_{i+1} = \overline{A}Q_i$ where $Q_{i+1} \in \mathbb{R}^{n\times n}$ is orthogonal and $R_{i+1} \in \mathbb{R}^{n\times n}$ is upper triangular.

Here we compute all the eigenvalues, so we work with $k = n$.

$$Q_{i+1}R_{i+1} = \overline{A}Q_i \;\Leftrightarrow\; \underbrace{Q_i^\mathsf{T}Q_{i+1}}_{:=\tilde{Q}_{i+1}} R_{i+1} = \underbrace{Q_i^\mathsf{T}\overline{A}Q_i}_{:=A_i}$$

By definition, $\tilde{Q}_{i+1}$ is orthogonal and A_i is similar to $\overline{A}$. So we have $\tilde{Q}_{i+1}R_{i+1} = A_i$. By swapping the factors, we get

$$R_{i+1}\tilde{Q}_{i+1} = (Q_{i+1}^\mathsf{T}\overline{A}Q_i)\cdot(Q_i^\mathsf{T}Q_{i+1}) = Q_{i+1}^\mathsf{T}\overline{A}Q_{i+1} = A_{i+1}.$$

QR algorithm.

1. Transform A to upper Hessenberg form, using left and right multiplication with Givens rotations. This gives $\overline{A} = A_0$, an upper Hessenberg matrix that has the same eigenvalues as A.
2. Compute for i=0,1,2,... the QR decomposition $\tilde{Q}_{i+1}R_{i+1} = A_i$ and swap the two factors: $A_{i+1} = R_{i+1}\tilde{Q}_{i+1}$.

Remark.
Every matrix A_i has upper Hessenberg form. Therefore we only need to create zeros at the subdiagonal of A_i (which means applying n-1 Givens rotations) to find its QR decomposition.
It can be shown that the subdiagonal of A_i converges to 0, which means that A_i converges to upper triangular form.
Another way to compute the QR algorithm is to use Householder transformations, which create zeros under the diagonal of A.

Definition 4.7 (Householder transformation).
Let $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq 0$. Then the matrix $H := I - 2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}}$ is called Householder transformation.

Proposition 4.11.
Let H be the Householder transformation. Then H is symmetric and orthogonal.

Proof.
$H^\mathsf{T} = \left(I - 2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}}\right)^\mathsf{T} = I^\mathsf{T} - \left(2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}}\right)^\mathsf{T} = I - 2 \cdot \frac{I}{\mathbf{x}^\mathsf{T}\mathbf{x}} \cdot (\mathbf{x}\mathbf{x}^\mathsf{T})^\mathsf{T} = I - 2 \cdot \frac{I}{\mathbf{x}^\mathsf{T}\mathbf{x}} \cdot \mathbf{x}\mathbf{x}^\mathsf{T}$,
so $H^\mathsf{T} = H$, which means that H is symmetric.
$H^\mathsf{T}H = HH = \left(I - 2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}}\right)\left(I - 2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}}\right) = I - 4 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}} + 4 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{x}\mathbf{x}^\mathsf{T}}{(\mathbf{x}^\mathsf{T}\mathbf{x})^2} =$
$= \mathrm{I} - 4 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}} + 4 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}} = I$, so $H^\mathsf{T} = H^{-1}$, therefore H is orthogonal. □

Interpretation of the Householder transformation as mapping.
Let $\mathbf{y} \in \mathbb{R}^n$. Then $H\mathbf{y} = \left(I - 2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}}{\mathbf{x}^\mathsf{T}\mathbf{x}}\right) \cdot \mathbf{y} = \mathbf{y} - 2 \cdot \frac{\mathbf{x}\mathbf{x}^\mathsf{T}\mathbf{y}}{\mathbf{x}^\mathsf{T}\mathbf{x}} = \mathbf{y} - 2 \cdot \frac{\mathbf{x}^\mathsf{T}\mathbf{y}}{\mathbf{x}^\mathsf{T}\mathbf{x}} \cdot \mathbf{x}$.
Let us take two special choices for $\mathbf{y}$. Let $\mathbf{y_1},\ \mathbf{y_2} \in \mathbb{R}^n$.
If $\mathbf{y_1}$ is parallel to $\mathbf{x}$ ($\mathbf{y_1} = c\mathbf{x}$), then $H\mathbf{y_1} = c\mathbf{x} - 2c \cdot \frac{<\mathbf{x},\mathbf{x}>}{<\mathbf{x},\mathbf{x}>} \cdot \mathbf{x} = -c\mathbf{x} = -\mathbf{y_1}$.
If $\mathbf{y_2}$ is orthogonal to $\mathbf{x}$, then $< \mathbf{x}, \mathbf{y_2} >= \mathbf{x}^\mathsf{T}\mathbf{y_2} = 0$, so $H\mathbf{y_2} = \mathbf{y_2}$.
Every vector $\mathbf{w} \in \mathbb{R}^n$ can be decomposed to a component that is parallel to $\mathbf{x}$ and one that is orthogonal (so $\mathbf{w}$ has the form $\mathbf{w} = \mathbf{y_1} + \mathbf{y_2}$). Therefore we have $H\mathbf{w} = H\mathbf{y_1} + H\mathbf{y_2}$ and we got that the Householder transformation is a reflection to the hyperplane $\mathbf{x}^\perp = \{\mathbf{y} \in \mathbb{R}^n :< \mathbf{x}, \mathbf{y} >= 0\}$.

The following proposition says that that any vector can be transformed to a special form with the suitable Householder transformation. So we can create zeros at all coordinates of the vector except the first one.

Proposition 4.12.
Let $\mathbf{y} \in \mathbb{R}^n$. There exists $\mathbf{x} \in \mathbb{R}^n$ with $\|\mathbf{x}\|_2 = 1$, such that $\mathrm{H}\mathbf{y} = \sigma e_1$, where $\sigma = \pm\|\mathbf{y}\|_2$ and $\mathbf{x} = \frac{\mathbf{y} - \sigma e_1}{\|\mathbf{y} - \sigma e_1\|_2}$.

Proof.
For the defined vector $\mathbf{x}$ it is obviously true that $\|\mathbf{x}\| = 1$. If $\mathbf{y} = 0$, then $\sigma = 0$ and the proposition holds.
Let us assume that $\mathbf{y} \neq 0$. By definition $< \mathbf{y}, \mathbf{y} >= \sigma^2$. Let us substitute $\mathbf{x}$ into the definition of the Householder matrix.

$$\left(I - 2 \cdot \frac{\mathbf{y} - \sigma e_1}{\|\mathbf{y} - \sigma e_1\|_2} \frac{(\mathbf{y} - \sigma e_1)^\mathsf{T}}{\|\mathbf{y} - \sigma e_1\|_2}\right)\mathbf{y} = \mathbf{y} - \frac{2}{\|\mathbf{y} - \sigma e_1\|_2^2}(\mathbf{y} - \sigma e_1)(< \mathbf{y}, \mathbf{y} > - \sigma < e_1, \mathbf{y} >)$$
$$= \mathbf{y} - (\mathbf{y} - \sigma e_1) = \sigma e_1$$

We use that $\|\mathbf{y} - \sigma e_1\|_2^2$ in the denominator is the same as
$< \mathbf{y} - \sigma e, \mathbf{y} - \sigma e_> = < \mathbf{y}, \mathbf{y} > -2\sigma < \mathbf{y}, e_1 > +\sigma^2 = 2(< \mathbf{y}, \mathbf{y} - \sigma < \mathbf{y}, e_1 >)$ and therefore we can simplify the fraction by this expression. □

Now we can define the QR algorithm by using Householder transformations.

QR algorithm.

Now we indicate the vector $\mathbf{x}$ in the Householder transformation ($H = H(\mathbf{x})$).
Let $A = [\mathbf{a_1}, \mathbf{a_2}, ..., \mathbf{a_n}] \in \mathbb{R}^{n \times n}$. Then according to the previous proposition $\exists \mathbf{x_1} \in \mathbb{R}^n$ with $\|\mathbf{x_1}\|_2 = 1$ such that $H(\mathbf{x_1})\mathbf{a_1} = \sigma_1 e_1$. We get that

$H(\mathbf{x_1})A = \begin{pmatrix} \sigma_1 & c_1^\mathsf{T} \\ 0 & A_1 \end{pmatrix}$, where $A_1 \in \mathbb{R}^{n-1 \times n-1}$ and $H_1 := H(\mathbf{x_1})$ is orthogonal.

Similarly for $A_1 = [\mathbf{a_1}^{(1)}, \mathbf{a_2}^{(1)}, ..., \mathbf{a_{n-1}}^{(1)}]$ there exists $\mathbf{x_2} \in \mathbb{R}^{n-1}$ with $\|\mathbf{x_2}\|_2 = 1$ such that $H(\mathbf{x_2})\mathbf{a_1}^{(1)} = \sigma_2 e_1$.

Then $H_2 := \begin{pmatrix} 1 & 0 \\ 0 & H(\mathbf{x_2}) \end{pmatrix}$ is also orthogonal and $H_2 H_1 A = \begin{pmatrix} \sigma_1 & c_1^\mathsf{T} & \\ 0 & \sigma_2 & c_2^\mathsf{T} \\ 0 & 0 & A_2 \end{pmatrix}$.

If we continue this method, in n-1 steps we will obtain an upper triangular matrix, which contains the values σ_1 in its diagonal. $H_{n-1}H_{n-2}...H_2H_1A = R$, and because H_i is symmetric and orthogonal, we have $A = \underbrace{H_1H_2...H_{n-2}H_{n-1}}_{Q:=} R = QR$.

4.3 Singular Value Decomposition

If $A \in \mathbb{R}^{m \times n}$ with $m \geq n$, then $A^\mathsf{T}A$ is a square real symmetric $n \times n$ matrix.

Theorem 4.13.
The eigenvalues of $A^\mathsf{T}A$ are nonnegative.

Proof. If λ is an eigenvalue of $A^\mathsf{T}A$, then $\exists\ \mathbf{u} \in \mathbb{R}^n$ such that $A^\mathsf{T}A\mathbf{u} = \lambda\mathbf{u}$. If we multiply both sides by $\mathbf{u}$ from the right, we get $< A^\mathsf{T}A\mathbf{u},\ \mathbf{u} > = \quad \lambda < \mathbf{u}, \mathbf{u} >$. This is equivalent to $< A\mathbf{u}, A\mathbf{u} > = \ \lambda < \mathbf{u}, \mathbf{u} >$. Rearranging this equation we obtain $\lambda = \frac{<A\mathbf{u}, A\mathbf{u}>}{<\mathbf{u}, \mathbf{u}>} \geq 0$. □

Definition 4.8 (Singular values).
Let us denote the eigenvalues of $A^\mathsf{T}A$ by $(\sigma_1)^2, ..., (\sigma_n)^2$. The numbers $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n \geq 0$ are called the singular values of A.

Let $\mathbf{v_i}$ be an eigenvector of $A^\mathsf{T}A$ that corresponds to the eigenvalue $(\sigma_i)^2$.

Then $A^\mathsf{T} A\mathbf{v_i} = (\sigma_i)^2\mathbf{v_i}$. The vectors $\mathbf{v_1}, ..., \mathbf{v_n}$ can be chosen such that they form an orthonormal system.

Definition 4.9 (Right singular vectors).
The vectors $\mathbf{v_1}, ..., \mathbf{v_n}$ defined above are called the right singular vectors of A.

If $A \in \mathbb{R}^{m\times n}$, then AA^T is an $m \times m$ real symmetric matrix.

We saw that $A^\mathsf{T} A\mathbf{v_i} = (\sigma_i)^2\mathbf{v_i}$. If we multiply both sides by A from the left, we obtain $AA^\mathsf{T} A\mathbf{v_i} = (\sigma_i)^2 A\mathbf{v_i}$. If A has independent columns, then $A\mathbf{v_i} \neq 0$, so $A\mathbf{v_i}$ is an eigenvector of AA^T, with corresponding eigenvalue $(\sigma_i)^2$.

Proposition: If we take $\mathbf{u_i} = \frac{1}{\sigma_i} A\mathbf{v_i}$, then $\mathbf{u_i}$ is an eigenvector, $\|\mathbf{u_i}\|_2 = 1$ and $< \mathbf{u_i}, \mathbf{u_j} >= 0$, if $i \neq j$.

Proof: We know that $A^\mathsf{T} A\mathbf{v_i} = (\sigma_i)^2\mathbf{v_i}$ and $< \mathbf{v_i}, \mathbf{v_j} > = \delta_{ij} \ = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$

If we take $\mathbf{u_i} \ = \ \frac{1}{\sigma_i} A\mathbf{v_i}$, we have

$$\begin{aligned} < \mathbf{u_i}, \mathbf{u_j} > \ &= < \frac{1}{\sigma_i} A\mathbf{v_i}, \ \frac{1}{\sigma_j} A\mathbf{v_j} > = \ \frac{1}{\sigma_i \sigma_j} < \mathbf{v_i}, A^\mathsf{T} A\mathbf{v_j} >= \\ &= \ \frac{1}{\sigma_i \sigma_j} \sigma_j^2 < \mathbf{v_i}, \mathbf{v_j} > = \ \frac{\sigma_j}{\sigma_i} \ \delta_{ij} \ = \ \delta_{\cdot} \end{aligned}$$ □

AA^T is an $m \times m$ matrix, so it has m eigenvalues. We have already found $(\sigma_1)^2, ..., (\sigma_n)^2$, with the corresponding eigenvectors $\mathbf{u_1}, ...\mathbf{u_n}$.
Let $\mathbf{z} \in \mathbb{R}^n, \ \mathbf{z} \neq 0$, such that $\mathbf{z}$ is orthogonal to every column of A.
Then

$$\begin{aligned} z \perp \mathbf{a_i} \quad \forall i = 1, 2, ..., n \\ \Leftrightarrow < \mathbf{z}, \mathbf{a_i} > = \ 0 \quad \forall i = 1, 2, ..., n \\ \Leftrightarrow \ \mathbf{a_i}^\mathsf{T} \mathbf{z} \ = \ 0 \quad \forall i = 1, 2, ..., n. \end{aligned}$$

It follows that $A^\mathsf{T}\mathbf{z} = 0$, and hence $AA^\mathsf{T}\mathbf{z} = 0$.
Now we have that $\mathbf{z}$ is an eigenvector with eigenvalue $\lambda = 0$. We can find $m - n$ linearly independent eigenvectors for the eigenvalue $\lambda = 0$.
Choose $\mathbf{u_{n+1}}, ..., \mathbf{u_m}$ such that $A^\mathsf{T} A\mathbf{u_i} = 0 \quad \forall i = n + 1, n + 2, ..., m$ and the vectors $\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_m}$ form an orthonormal system.
So we obtained all the eigenvalues and eigenvectors of AA^T:
The eigenvalues are $(\sigma_1)^2, (\sigma_2)^2, ..., (\sigma_n)^2, 0, 0, ..., 0$.
The eigenvectors are $\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_m}$, and they form an orthonormal system where

$$AA^\mathsf{T}\mathbf{u_i} = \begin{cases} (\sigma_i)^2\mathbf{u_i} & (i = 1, 2, ..., n) \\ 0 & (i = n + 1, n + 2, ..., m) \end{cases}$$

Definition 4.10 (Left singular vectors).
The vectors $\mathbf{u_1}, ..., \mathbf{u_m}$ defined above are called the left singular vectors of A.

Theorem 4.14 (Singular Value Decomposition).
Let $A \in \mathbb{R}^{m\times n}$, with $m \geq n$. Then there exist orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\Sigma = \text{diag}(\sigma_1, ..., \sigma_n) \in \mathbb{R}^{m\times n}$ such that

$$A = U\Sigma V^{\mathsf{T}}.$$

(For proof see [11] page 270-271.)

4.3.1 Application of SVD: Approximating a matrix

Theorem 4.15 (Decomposition into rank-1 matrices).
Any matrix A can be decomposed as a sum of rank-1 matrices:

$$A = \sigma_1 \mathbf{u_1}\mathbf{v_1}^{\mathsf{T}} + \sigma_2 \mathbf{u_2}\mathbf{v_2}^{\mathsf{T}} + \sigma_n \mathbf{u_n}\mathbf{v_n}^{\mathsf{T}}.$$

Here $(\mathbf{u_i})$ and $(\mathbf{v_i})$ are the column vectors of the orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$, respectively, and the numbers (σ_i) are the diagonal entries of $\Sigma \in \mathbb{R}^{m\times n}$ from the Singular Value Decomposition.
We have $\|\mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}\|_2 = 1$.

Proof.
The decomposition follows from the construction of the vectors $(\mathbf{u_i})$, $(\mathbf{v_i})$ and the definition of singular values. We need to prove that $\|\mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}\|_2 = 1$.
Recall: For any matrix $M \in \mathbb{R}^{n\times n}$: $\|M\|_2 = \max_{\mathbf{x}\neq 0} \frac{\|M\mathbf{x}\|_2}{\|\mathbf{x}\|_2} = \max_{\|\mathbf{y}\|_2=1} \|M\mathbf{y}\|_2$. Now compute this norm for $M = \mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}$.

$$\begin{aligned}
\|\mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}\|_2 &= \max_{\|\mathbf{y}\|_2=1} \|\mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}\mathbf{y}\|_2 \\
&= \max_{\|\mathbf{y}\|_2=1} |\mathbf{v_i}^{\mathsf{T}}\mathbf{y}| \cdot \|\mathbf{u_i}\|_2 && (10)\\
&= \max_{\|\mathbf{y}\|_2=1} \|\mathbf{v_i}\|_2 \cdot \|\mathbf{y}\|_2 \cdot |\cos(\alpha)| && (11)\\
&= 1 && (12)
\end{aligned}$$

To obtain equality (10) we use that $\mathbf{v_i}^{\mathsf{T}}\mathbf{y}$ is a constant, so we can bring it out from the norm, taking absolute value. For getting (11), we first recognize that $\|\mathbf{u_i}\|_2 = 1$, because it is a column vector of the orthogonal matrix U, used in the SVD decomposition. Second, we substitute in the definition of dot product, which gives us $|\mathbf{v_i}^{\mathsf{T}}\mathbf{y}| \cdot \|\mathbf{u_i}\|_2 = \|\mathbf{v_i}\|_2 \cdot \|\mathbf{y}\|_2 \cdot |\cos(\alpha)|$. For the equality (12), we use that $\|\mathbf{y}\|_2 = 1$ by definition, and $\|\mathbf{v_i}\|_2 = 1$, because it is a column vector of the

orthogonal matrix V from the SVD. We know that the maximum of $|cos(\alpha)|$ is 1, and now we are done. □

The SVD method can be used to approximate matrices as follows.
Define $\Sigma_k = \text{diag}(\sigma_1, \sigma_2, ..., \sigma_k, 0, 0, ..., 0)$ for $k \leq n$.
The matrix

$$A_k \;=\; U\Sigma_k V^{\mathsf{T}} \;=\; \sigma_1 \mathbf{u_1}\mathbf{v_1}^{\mathsf{T}} + \sigma_2 \mathbf{u_2}\mathbf{v_2}^{\mathsf{T}} + ... + \sigma_k \mathbf{u_k}\mathbf{v_k}^{\mathsf{T}}$$

is a rank-k approximation of A. Furthermore, this A_k matrix is the best rank-k approximation of A, when the error is measured in either the 2-norm, or the Frobenius norm. Here we will only show this property in the 2-norm. For the Frobenius norm, see [4], page 121-122.

Lemma.

$$\|A - A_k\|_2^2 = \sigma_{k+1}^2$$

Proof.
Let $A = \sum_{i=1}^{n} \sigma_i \mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}$ be the singular value decomposition of A. Then
$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}$ and $A - A_k = \sum_{i=k+1}^{n} \sigma_i \mathbf{u_i}\mathbf{v_i}^{\mathsf{T}}$. Let $\mathbf{v}$ be the top singular vector of $A - A_k$ (the one associated with the largest singular value). If we express $\mathbf{v}$ as a linear combination of $\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_n}$, we get $\mathbf{v} = \sum_{i=1}^{n} \alpha_i \mathbf{v_i}$. Then

$$|(A - A_k)\mathbf{v}| \;=\; \left| \sum_{i=k+1}^{n} \sigma_i \mathbf{u_i}\mathbf{v_i}^{\mathsf{T}} \sum_{j=1}^{n} \alpha_j \mathbf{v_j} \right| \;=\; \left| \mathbf{v_i} \sum_{i=k+1}^{n} \alpha_i \sigma_i \mathbf{u_i}\mathbf{v_i}^{\mathsf{T}} \mathbf{v_i} \right|$$

$$= \left| \sum_{i=k+1}^{n} \alpha_i \sigma_i \mathbf{u_i} \right| \;=\; \sqrt{\sum_{i=k+1}^{n} \alpha_i^2 \sigma_i^2}$$

The vector $\mathbf{v}$ maximizing this last quantity, subject to the constraint that $|\mathbf{v}|^2 = \sum_{i=1}^{n} \alpha_i^2 = 1$, occurs when $\alpha_{k+1} = 1$ and the rest of the α_i are 0. Therefore, $\|A - A_k\|_2^2 = \sigma_{k+1}^2$. □

Now we will prove that A_k is the best rank-k approximation of A.

Theorem 4.16.
Let $A \in \mathbb{R}^{m \times n}$. For any matrix B of rank at most k,

$$\|A - A_k\|_2 \leq \|A - B\|_2.$$

Proof.
If A is of rank k or less, the theorem is obviously true since $\|A - A_k\|_2 = 0$.

Hence, assume that A is of rank greater than k. From the Lemma above, we know that

$$\|A - A_k\|_2^2 = \sigma_{k+1}^2.$$

Indirectly, let us suppose that there is some matrix B of rank at most k such that B is a better 2-norm approximation of A than A_k. This means that

$$\|A - B\|_2 < \sigma_{k+1}.$$

The null space of B, Null(B), has dimension at least n-k. Let $\mathbf{v_1}, \mathbf{v_2}, ...\mathbf{v_{k+1}}$ be the first k+1 singular vectors of A. By dimension argument, it follows that there exists a vector $\mathbf{z} \neq 0$, that $\mathbf{z} \in \text{Null}(B) \cap \text{span}\{\mathbf{v_1}, \mathbf{v_2}, ..., \mathbf{v_{k+1}}\}$. Let us scale $\mathbf{z}$ such that $|\mathbf{z}| = 1$. We know that

$$\|A - B\|_2^2 \geq |(A - B)\mathbf{z}|^2.$$

Now use that $\mathbf{z} \in \text{Null}(B)$, which implies $B\mathbf{z} = 0$, and get

$$\|A - B\|_2^2 \ \geq \ |A\mathbf{z}|^2.$$

Since $\mathbf{z} \in \text{span}\{\mathbf{v_1}, ..., \mathbf{v_{k+1}}\}$, we have

$$\begin{aligned} |A\mathbf{z}|^2 \ &= \ \left| \sum_{i=1}^{m} \sigma_i \mathbf{u_i} \mathbf{v_i}^\mathsf{T} \mathbf{z} \right|^2 \ = \ \sum_{i=1}^{m} \sigma_i^2 (\mathbf{v_i}^\mathsf{T} \mathbf{z})^2 \ = \ \sum_{i=1}^{k+1} \sigma_i^2 (\mathbf{v_i}^T \mathbf{z})^2 \\ &\geq \ \sigma_{k+1}^2 \sum_{i=1}^{k+1} (\mathbf{v_i}^\mathsf{T} \mathbf{z})^2 \ = \ \sigma_{k+1}^2. \end{aligned}$$

It follows that $\|A - B\|_2^2 \geq \sigma_{k+1}^2$. These leads to a contradiction, because we assumed that $\|A - B\|_2^2 < \sigma_{k+1}^2$. □

4.3.2 Solving ill-conditioned systems using SVD

Since now we know what is the Singular Value Decomposition, I would like to present the article which uses the SVD to solve ill-conditioned systems efficiently [10].

Consider the system

$$A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{n \times n}, \quad \mathbf{b} \in \mathbb{R}^n \tag{13}$$

in case it is ill-conditioned. Here we approach the problem as a least squares problem, because it allows us to use more kind of methods, such as the Singular Value Decomposition. The weak spot of this method is calculating the small singular values, therefore we take the truncated version of the SVD (TSVD), neglecting the dangerous singular values that could cause a problem. After we create a new system with a reduced condition number, the problem is approached with Gauss elimination.

The SVD of the coefficients' matrix takes the form

$$A = V\Sigma U^{\mathsf{T}} = \sum_{i=1}^{m} \sigma_i \mathbf{v_i} \mathbf{u_i}^{\mathsf{T}}, \tag{14}$$

where Σ is an $m \times m$ diagonal matrix of singular values $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_m \geq 0$ and V and U are square matrices formed of the left ($\mathbf{u_i}$) and right ($\mathbf{v_i}$) singular vectors. By using (14), for the solution of (13) we get

$$\mathbf{x} = \sum_{i=1}^{m} \sigma_i^{-1} \mathbf{u_i} (\mathbf{v_i}^{\mathsf{T}} \mathbf{b}). \tag{15}$$

Even though the small singular values are inherent in the SVD of badly conditioned matrices, they are often computed inaccurate with this method. Consequently, the solution (15) may not be accurate.
Intuition suggests that existing algorithms collapse in solving ill-conditioned problems at some stage, where "nearness' of the initial coefficient matrix columns (rows) becomes crucial. The algorithm presented here is an attempt to provide an accurate solution to the problem.

The algorithm.

Let the following "dangerous" small singular values be neglected in the SVD solution:

$$\epsilon \geq \sigma_{n+1} \geq \sigma_{n+2} \geq ... \geq \sigma_m.$$

Then (15) takes the form

$$\mathbf{x_1} = \sum_{i=1}^{n} \sigma_i^{-1} \mathbf{u_i} (\mathbf{v_i}^{\mathsf{T}} \mathbf{b}). \tag{16}$$

The parameter ϵ defines the truncated version of the SVD (TSVD). The TSVD solution (16) is widely used as a regularized LS problem solution. For the purpose of this paper, however, solution (16) is not accurate enough. It may be pin-pointed in the following way.

Let the matrices of the left and right singular vectors of the TSVD be designated as $V_1 = [\mathbf{v_1}...\mathbf{v_n}]$ and $U_1 = [\mathbf{u_1}...\mathbf{u_n}]$. Then their orthogonal complements are $\tilde{V_2} = [\tilde{\mathbf{v}}_{\mathbf{n+1}}...\tilde{\mathbf{v}}_{\mathbf{m}}] \equiv (V_1)^{(\perp)}$ and $\tilde{U_2} = [\tilde{\mathbf{u}}_{\mathbf{n+1}}...\tilde{\mathbf{u}}_{\mathbf{m}}] \equiv U_1^{(\perp)}$, correspondingly. Columns of matrices $\tilde{V_2}$ and $\tilde{U_2}$ span nullspaces of matrices V_1^{T} and U_1^{T}. Instead of using singular vectors of the full SVD corresponding to neglected singular values, it is preferable to compute orthonormal sets $\tilde{\mathbf{v}}_{\mathbf{n+1}}...\tilde{\mathbf{v}}_{\mathbf{m}}$ and $\tilde{\mathbf{u}}_{\mathbf{n+1}}...\tilde{\mathbf{u}}_{\mathbf{m}}$ on the base of "good " matrices $(V_1)^{\mathsf{T}}$ and $(U_1)^{\mathsf{T}}$. It may be carried out in various ways as shown in the next section.

Let us split the unknowns in (14) as follows:

$$\mathbf{x} = U_1\mathbf{z_1} + \tilde{U}_2\mathbf{z_2}.$$

If we multiply (13) by $[V_1, V_2]^\mathsf{T}$ from the left, (13) takes the form

$$\tilde{A}\mathbf{z} = \begin{bmatrix} \text{diag}(\sigma_1, ..., \sigma_n) & 0 \\ 0 & C \end{bmatrix} \begin{bmatrix} \mathbf{z_1} \\ \mathbf{z_2} \end{bmatrix} = \begin{bmatrix} \mathbf{b_1} \\ \mathbf{b_2} \end{bmatrix}, \tag{17}$$

where

$$\begin{aligned} \tilde{A} &= [V_1, \tilde{V}_2]^\mathsf{T} A [U_1, \tilde{U}_2], \\ C &= \tilde{V}_2^\mathsf{T} A \tilde{U}_2, \\ \mathbf{b_1} &= V_1^\mathsf{T}\mathbf{b}, \qquad \mathbf{b_2} = \tilde{V}_2^\mathsf{T}\mathbf{b}. \end{aligned}$$

The new system of linear equations (17) can be solved independently for $\mathbf{z_1}$ and $\mathbf{z_2}$. The vector $\mathbf{z_1}$ corresponds to the TSVD solution discussed above,

$$U_1\mathbf{z_1} = \mathbf{x_1}$$

and the vector $\mathbf{z_2}$ is computed from the following equation:

$$C\mathbf{z_2} = \mathbf{b_2}.$$

4.4 Spectrum slicing

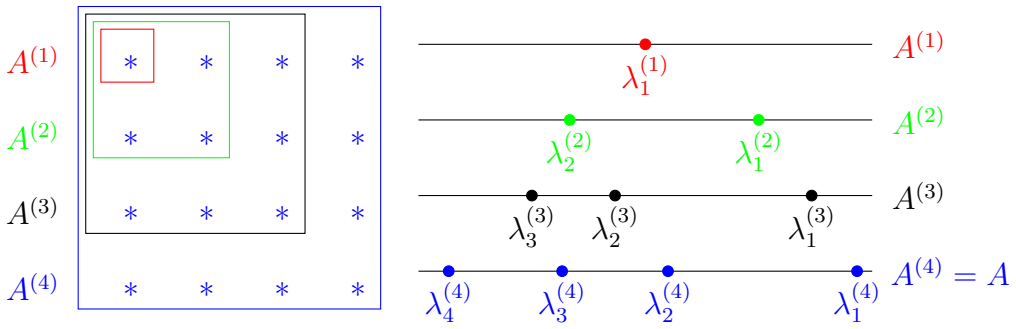
We want to compute one or more eigenvalues of a symmetric matrix. For this we first need to get familiar with the Sturm sequences and the Cauchy interlacing theorem.

4.4.1 Cauchy interlacing theorem

Theorem 4.17 (Cauchy interlacing theorem).
Let $A \in \mathbb{R}^{n\times n}$ be symmetric. Let $A^{(j)}, \quad (j = 1, ..., n)$ be the j^{th} order principal leading matrices, namely the $j \times j$ upper left blocks of A.
Then the eigenvalues of $A^{(j)}$ interlace the eigenvalues of $A^{(j+1)}$.

Remark.
This means that if $A^{(j)}$ has the eigenvalues $\lambda_k,\ k = 1, 2, ..., j$, with
$\lambda_1^{(j)} \geq \lambda_2^{(j)} \geq ... \geq \lambda_{j-1}^{(j)} \geq \lambda_j^{(j)}, \quad$ then: $\lambda_{k+1}^{(j+1)} \leq \lambda_k^{(j)} \leq \lambda_k^{(j+1)}, \quad k = 1, 2, ..., j.$

For example, for $n = 4$, we can have the following:



To prove this theorem, let us see first the Courant-Fischer theorem.

Theorem 4.18 (Courant-Fischer theorem).
Let $A \in \mathbb{R}^{n\times n}$ symmetric. Denote the eigenvalues of A as $\lambda_1, \lambda_2, ..., \lambda_n$ with $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$. Then

$$\lambda_k = \min_{\{S\subseteq\mathbb{R}^n,\ \dim S = n-k+1\}} \max_{\mathbf{x}\in S, \mathbf{x}\neq 0} \frac{< A\mathbf{x},\ \mathbf{x} >}{< \mathbf{x},\ \mathbf{x} >}$$

for $k = 1, 2, ..., n$.

Proof. Let us choose the eigenvectors $\mathbf{u_1}, \mathbf{u_2}, ...\mathbf{u_n}$ of A such that they form an orthonormal system.
Let $k \in \{1, 2, ..., n\}$. Define the space $S_k := \text{ span}\{\mathbf{u_1}, \mathbf{u_2}, ..., \mathbf{u_k}\}$. Here $\dim S_k = k$. We will prove the equality in the theorem by showing that inequality holds in both directions.

(1) First, we show for every $S \subseteq \mathbb{R}^n$ with $\dim S = n - k + 1$, that

$$\max_{\mathbf{x}\in S, \mathbf{x}\neq 0} \frac{< A\mathbf{x},\ \mathbf{x} >}{< \mathbf{x},\ \mathbf{x} >} \geq \lambda_k.$$

Let $S \subseteq \mathbb{R}^n,\ \dim S = n - k + 1$. Since $\dim S_k = k$, we know that $S \cap S_k \neq \emptyset$. For this reason there exists $\mathbf{x} \in \mathbb{R}^n,\ \mathbf{x} \neq 0$, such that $\mathbf{x} \in S \cap S_k$. Because $\mathbf{x} \in S_k$, we can write $\mathbf{x}$ as the linear combination of the basis vectors of S_k : $\mathbf{x} = \sum_{j=1}^{k} \beta_j \mathbf{u_j}$.

Then we have $\frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>} = \frac{\sum_{j=1}^{k} \beta_j^2\ \lambda_j}{\sum_{j=1}^{k} \beta_j^2} \geq \frac{\sum_{j=1}^{k} \beta_j^2\ \lambda_k}{\sum_{j=1}^{k} \beta_j^2} = \lambda_k.$

(2) Now we construct a space $S \subseteq \mathbb{R}^n$ with $\dim S = n - k + 1$, such that $\frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>} \leq \lambda_k$.
Define $S = \text{span}\{\mathbf{u_k}, \mathbf{u_{k+1}}, ..., \mathbf{u_n}\}$. Here $\dim S = n - k + 1$, therefore the intersection of S and S_k is not empty. Again, let $\mathbf{x} \in S \cap S_k,\ \mathbf{x} \neq 0$.

Writing $\mathbf{x}$ in the basis of S, we have $\mathbf{x} = \sum_{j=k}^{n} \gamma_j \mathbf{u_j}$.

If we use this form of $\mathbf{x}$, we obtain $\frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>} = \frac{\sum_{j=k}^{n} \gamma_j^2\ \lambda_j}{\sum_{j=k}^{n} \gamma_j^2} \leq \frac{\sum_{j=k}^{n} \gamma_j^2\ \lambda_k}{\sum_{j=k}^{n} \gamma_j^2} = \lambda_k.$ □

Now we can prove the Cauchy interlacing theorem.

Proof (Cauchy interlacing theorem)

We will show the property for $j = n - 1$. We need to prove that

$$\lambda_{k+1}^{(j+1)} \leq \lambda_k^{(j)} \leq \lambda_k^{(j+1)}.$$

The Courant-Fischer theorem gives the following results for $\lambda_k^{(n)}$ and $\lambda_k^{(n-1)}$:

$$\lambda_k^{(n)} = \min_{\{S \subseteq \mathbb{R}^n,\ \dim S = n-k+1\}}\ \max_{\mathbf{x} \in S,\ \mathbf{x} \neq 0} \frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>}$$

$$\lambda_k^{(n-1)} = \min_{\{S \subseteq \mathbb{R}^{n-1},\ \dim S = n-k\}}\ \max_{\mathbf{y} \in S,\ \mathbf{y} \neq 0} \frac{<A^{(n-1)}\ \mathbf{y},\ \mathbf{y}>}{<\mathbf{y},\ \mathbf{y}>}.$$

Let us define for $\mathbf{y} \in \mathbb{R}^{n-1}$ the vector $\mathbf{x} = \begin{pmatrix} \mathbf{y} \\ x_n \end{pmatrix} \in \mathbb{R}^n$ and add the requirement $x_n = 0$. By adding this extra requirement, we have

$$\lambda_k^{(n-1)} = \min_{\{S \subseteq \mathbb{R}^n,\ \dim S = n-k+1\}}\ \max_{\{\mathbf{x} \in S,\ \mathbf{x} \neq 0,\ x_n = 0\}} \frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>}.$$

Now we can see that $\lambda_k^{(n)} \geq \lambda_k^{(n-1)}$.

For $\lambda_{k+1}^{(n)}$ the Courant-Fischer theorem gives

$$\lambda_{k+1}^{(n)} = \min_{\{S \subseteq \mathbb{R}^n,\ \dim S = n-k\}}\ \max_{\mathbf{x} \in S,\ \mathbf{x} \neq 0} \frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>}.$$

We saw that

$$\lambda_k^{(n-1)} = \min_{\{S \subseteq \mathbb{R}^n,\ \dim S = n-k+1\}}\ \max_{\{\mathbf{x} \in S,\ \mathbf{x} \neq 0,\ x_n = 0\}} \frac{<A\mathbf{x},\ \mathbf{x}>}{<\mathbf{x},\ \mathbf{x}>}.$$

While this gives us $k-1$ general requirements and one special requirement ($x_n = 0$), the formula of $\lambda_{k+1}^{(n)}$ gives k general requirements.

Therefore we get $\lambda_k^{(n-1)} \geq \lambda_{k+1}^{(n)}$. □

4.4.2 Sturm sequences

Let p_j be the characteristic polynomial of $A^{(j)}$, so

$$p_0(\lambda) := 1$$
$$p_j(\lambda) := \det(A^{(j)} - \lambda I) \quad \text{for } j = 1, 2, ..., n.$$

Theorem 4.19 (Sturm's theorem).
Let $A \in \mathbb{R}^{n \times n}$ be symmetric and $\mu \in \mathbb{R}$.
Consider the sequence $(p_0(\mu),\ p_1(\mu), ..., p_n(\mu))$.
Let $s(\mu)$ be the number of sign agreements in the sequence.
Then $s(\mu)$ is the number of eigenvalues λ of A with $\lambda > \mu$.
Here if $p_j(\mu) = 0$, we assume a sign change.

Combining Sturm's theorem and Bisection method.
We would like to find the eigenvalue λ_i of A.
Basically we use the numbers $s(\mu)$ instead of the $f(x)$ function values from the Bisection method.

Step 1. First, we need to find $a,\ b \in \mathbb{R}$, such that $s(a) \geq i$ and $s(b) < i$. Sturm's theorem tells us that at least i eigenvalues are located to the right of a and less than i eigenvalues are located to the right of b. For this reason we know that $\lambda_i \in (a, b)$.

Step 2. Now we take the midpoint $m = \frac{a+b}{2}$ of the interval, and we compute $s(m)$.

$$\text{If } s(m) \begin{cases} \geq i, & \text{then} \quad a := m \\ < i, & \text{then} \quad b := m \end{cases}.$$

This way we bisected the interval, and it still contains λ_i.

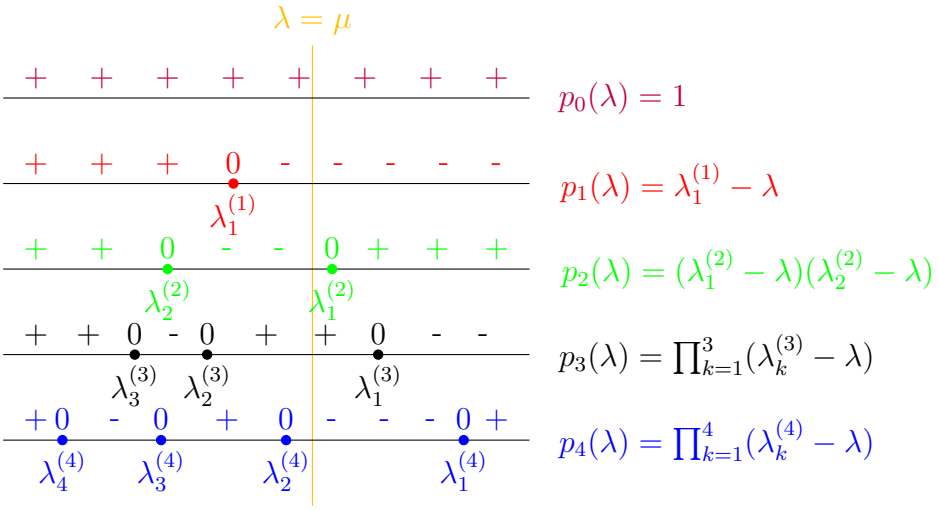
Step 3. If $b - a \leq \epsilon$ for a given $\epsilon > 0$, we obtained the solution with the required accuracy, so we stop. Otherwise, return to *Step 2.*.

Sturm sequences and eigenvalues.
The characteristic polynomials of $A^{(j)}$ are

$$p_j(\lambda) = (\lambda_1^{(j)} - \lambda) \cdot (\lambda_2^{(j)} - \lambda) \cdot ... \cdot (\lambda_j^{(j)} - \lambda).$$

From the Cauchy interlacing theorem we know that the eigenvalues of $A^{(j)}$ interlace the eigenvalues of $A^{(j+1)}$. This means that the zeros of p_j interlace the zeros of p_{j+1}.

Choose $\mu \in \mathbb{R}$ and simply count the number of sign agreements in the Sturm sequence.



In this example we find + - - + - , so there is one sign agreement. This means that $A = A^{(4)}$ has one eigenvalue that is bigger than μ.

Spectrum slicing method for symmetric matrices.

First, we transform A to an upper Hessenberg matrix with Givens rotations (see Definition *4.5*). Since A is symmetric, the result is a tridiagonal matrix.
Let us assume that

$$A = \begin{pmatrix} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & & \beta_n & \alpha_n \end{pmatrix}.$$

We would like to find the eigenvalue λ_i.
Let $s(\lambda)$ denote the number of sign agreements in the Sturm sequence. The Sturm sequence can be computed with the following recursion:

$$p_j(\lambda) = (\alpha_j - \lambda) \cdot (p_{j-1}(\lambda)) - \beta_j^2 \cdot p_{j-2}(\lambda), \quad j = 1, 2, ...$$

Define $p_{-1}(\lambda) = 0,\ p_0(\lambda) = 1.$

Now we use the previously introduced algorithm:

Step 1. First find $a,\ b \in \mathbb{R}$, such that $s(a) \geq i$ and $s(b) < i$.

Step 2. Take the midpoint $m = \frac{a+b}{2}$ of the interval, and compute $s(m)$.

$$\text{If } s(m) \begin{cases} \geq i, & \text{then} \quad a := m \\ < i, & \text{then} \quad b := m \end{cases}.$$

Step 3. If $b - a \leq \epsilon$ for a given $\epsilon > 0$, we obtained the solution with the required accuracy, so we stop. Otherwise, return to *Step 2.*

References

[1] brusspup. *Amazing Resonance Experiment!* 2013. URL: https://www.youtube.com/watch?v=wvJAgrUBF4w&feature=youtu.be.

[2] Martin J. Gander and Felix Kwok. "Chladni Figures and the Tacoma Bridge: Motivating PDE Eigenvalue Problems via Vibrating Plates." In: *SIAM Review* 54.3 (2012), pp. 573–596. DOI: 10.1137/10081931X.

[3] Amos Gilat and Vish Subramaniam. *Numerical methods for engineers and scientists, 3rd edition*. Wiley, 2013.

[4] John Hopcroft and Ravindran Kannan. "Foundations of Data Science". In: Cornell University, 2011. Chap. 4.

[5] Yuka Kobayashi and Takeshi Ogita. "A fast and efficient algorithm for solving ill-conditioned linear systems." In: *JSIAM Letters* 7 (2015), pp. 1–4.

[6] Gergó Lajos. *Numerikus módszerek*. ELTE Eötvös Kiadó, 2010.

[7] Carl Dean Meyer. *Matrix Analysis and Applied Linear Algebra*. Society for Industrial and Applied Mathematics, 2000.

[8] A. Nemirovski. "Optimization II. Numerical Methods for Nonlinear Continuous Optimization". In: Technion, Haifa, 1999. Chap. 2.

[9] Roger D. Peng. "Advanced Statistical Computing". In: (Work in progress), 2018. Chap. 2.

[10] Yu. Volokh and O. Vilnayk. "Pin-Pointing Solution of Ill-Conditioned Square Systems of Linear Equations." In: *Pergamon, Applied Mathematics Letters* 13 (2000), pp. 119–124.

[11] Martin J. Gander Walter Gander and Felix Kwok. *Scientific Computing*. Texts in Computational Science and Engineering 11. Springer, 2014.

[12] Jinmei Wang and Ke Wang. "Multi-scale method for ill-conditioned linear systems." In: *ICAMMT, IOP Conference Series: Materials Science and Engineering* 242 (2017), pp. 1–4. DOI: 10.1088/1757-899X/242/1/012100.

[13] Todd Young and Martin J. Mohlenkamp. *Introduction to Numerical Methods and Matlab Programming for Engineers*. Ohio University, 2018.

A Appendix

A.1 Gauss elimination

```
function x = Gauss(A,b)
%Input variables:
%   A: Coefficient matrix
%   b: Right hand side vector
%Output variable:
%   x: Solution of the linear system of equations
ab = [A,b];
[R, C] = size(ab);
%Gauss elimination procedure:
for j = 1:R - 1
    for i = j + 1:R
        ab(i,j:C) = ab(i,j:C) - ab(i,j)/ab(j,j)*ab(j,j:C);
    end
end
%Back substitution:
x = zeros(R,1);
x(R) = ab(R,C)/ab(R,R);
for i = R-1:-1:1
    x(i) = (ab(i,C) - ab(i,i+1:R)*x(i+1:R))/ab(i,i);
end
x
end
```

A.2 Gauss elimination with partial pivoting and scaling

```
function x = GaussPivotLarge(A,b)
%Input variables:
%   A: Coefficient matrix
%   b: Right hand side vector
%Output variable:
%   x: Solution of the linear system of equations
ab = [A,b];
[R, C] = size(ab);
% Pivoting section
for j = 1:R - 1
max=abs(ab(j,j));
maxi=j;
for k = j + 1:R
    if abs(ab(k,j)) > max
        max = abs(ab(k,j));
        maxi = k;
    end
end
abTemp = ab(j,:);
ab(j,:) = ab(maxi,:);
ab(maxi,:) = abTemp;
% Elimination part
for i = j + 1:R
    ab(i,j:C) = ab(i,j:C) - ab(i,j)/ab(j,j)*ab(j,j:C);
end
end
x = zeros(R,1);
x(R) = ab(R,C)/ab(R,R);
for i = R - 1:-1:1
    x(i) = (ab(i,C) - ab(i,i + 1:R)*x(i + 1:R))/ab(i,i);
end
x
end
```

A.3 Gauss-Jordan method

```
function x = GaussJordan(A,b)
%Input variables:
%   A: Coefficient matrix
%   b: Right hand side vector
%Output variable:
%   x: Solution of the linear system of equations
ab = [A,b];
[R, C] = size(ab);
%forward Gauss:
for j = 1:R - 1
%pivoting starts
max=abs(ab(j,j));
maxi=j;
for k = j + 1:R
    if abs(ab(k,j)) > max
        max = abs(ab(k,j));
        maxi = k;
    end
end
abTemp = ab(j,:);
ab(j,:) = ab(maxi,:);
ab(maxi,:) = abTemp;
%pivoting ends
    ab(j,j:C) = ab(j,j:C)/ab(j,j); %normalizing ab(j,j)
    for i = j + 1:R
        ab(i,j:C) = ab(i,j:C) - ab(i,j)*ab(j,j:C);
    end
end
%normalizing ab(R,R) by dividing the last row with ab(R,R):
ab(R,C-1:C) = ab(R,C-1:C)/ab(R,R);
%backward Gauss:
for j = R:-1:2
    for i = j - 1:-1:1
        ab(i,C:-1:j) = ab(i,C:-1:j) - ab(i,j)*ab(j,C:-1:j);
    end
end
%Now x equals to b, and b is the last column of ab, so we can get x
%by taking the last column of ab:
x=ab(:,C)
```

A.4 Gauss elimination using LU decomposition

```
function x = LUdecompGauss(A,b)
%Input variables:
%   A: Coefficient matrix
%   b: Right hand side vector
%Output variable:
%   x: Solution of the linear system of equations
[R, C] = size(A);
%Computing LU factorization
for j = 1:R - 1
    for i = j + 1:R
        L(i,j) = A(i,j)/A(j,j);
        A(i,j:C) = A(i,j:C) - A(i,j)/A(j,j)*A(j,j:C);
    end
end
L(:,C)=zeros(R,1);
L=tril(L,1)+eye(size(A));
U=triu(A);
LU=L*U;
%Solving Ly=b with forward substitution
Ly=b;
y(1)=L(1,1)/b(1);
for j = 2:R
    y(j) = (b(2)-L(i,1:i-1).*y(1:i-1))/L(i,i);
end
%Solving Ux=y with backward substitution
Ux=y;
x(R)=y(R)/U(R,R);
for i = R-1:-1:1
    x(i) = (y(i)-U(i,i+1:R).*x(i+1:R))/U(i,i);
end
x
end
```

A.5 Bisection method

```
function Xs = BisectionRoot(F,a,b,Err,imax)
% Input variables:
%   F: Name of a user-defined function that calculates F for
%      a given x
%   a,b: Two points in the neighborhood of the root
%        (on either side or the same side of the root)
%   Err: Maximum error
%   imax: Maximum number of iterations
% Output variable:
%   Xs: Solution
if F(a)*F(b)>0
   disp('Error: the given points are on the same side of the sol.')
else
   for i=1:imax
      xNS=(a+b)/2;
      Erri=(b-a)/2;
      if Erri<Err
          Xs=xNS; %found the solution within the required tolerance
          fprintf('The root of this nonlin. eq. is X=%11.6f\n',Xs)
          break;
      end
      if i==imax
          fprintf('Sol. was not found in %i iterations.\n',imax)
      end
      if F(a)*F(xNS)<0
          b=xNS;
      else
          a=xNS;
      end
   end
end
end
```

A.6 Newton's method

```
function Xs = NewtonSol(F,FDer,Xest,Err,imax)
% Input variables:
%   F: Name of a user-defined function that calculates F for
%      a given x
%   FDer: The derivative of F
%   Xest: The initial guess for the solution
%   Err: Maximum error
%   imax: Maximum number of iterations
% Output variable:
%   Xs: Solution
for i = 1:imax
    Xi = Xest - F(Xest)/FDer(Xest);
    if abs((Xi - Xest)/Xest) < Err
        Xs = Xi;
        fprintf('The root of the given equation is X=%11.6f. \n',Xs)
        break
    end
    Xest = Xi;
end
if i==imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
    Xs = ('No answer');
end
end
```

A.7 Secant method

```
function Xs = SecantRoot(F,a,b,Err,imax)
% Input variables:
%   F: Name of a user-defined function that calculates F for
%      a given x
%   a,b: Two points in the neighborhood of the root
%        (on either side or the same side of the root)
%   Err: Maximum error
%   imax: Maximum number of iterations
% Output variable:
%   Xs: Solution
for i=1:imax
    Xi = b - ((F(b)*(a-b))/(F(a)-F(b)));
    if abs((Xi - b)/b) < Err
        Xs=Xi; %Solution is found within the desired tolerance.
        break;
    end
    a = b;
    b = Xi;
end
if i==imax
    fprintf('Solution was not obtained in %i iterations.\n',imax)
end
```