Two-level preconditioning applied on the ssSNPBLUP model

by

Buu-Van Nguyen

to obtain the degree of Master of Science
in Applied Mathematics
at the Delft University of Technology
Faculty of Electrical Engineering, Mathematics and Computer Science,
to be defended publicly on Tuesday July 19, 2021 at 15:00.

Student number:	4218728	
Project duration:	July 22, 2020 – July 19, 2021	
Thesis committee:	Prof. dr. ir. C. Vuik,	TU Delft, supervisor
	Dr. Ir. J. Vandenplas,	WUR, DLO Researcher, supervisor
	Prof. Dr. Ir. H. X. Lin,	TU Delft

Acknowledgement

It has been an odd time during the whole COVID-19 pandemic. It took some time for me to adjust to a working at home lifestyle. It may sound freakish, but I have not seen my supervisors in person during the MSc Thesis project. However, my supervisors *Kees Vuik* and *Jérémie Vandenplas* have done a fantastic job in supporting and guiding me through this project. Therefore, I would like to show my appreciation to them with some kind words. They are able to create a friendly atmosphere. Moreover, the patience and the time these people take for their students, makes me feel quite privileged to be able to work with them. Hence I owe you a big thank you.
My family especially my little brother *Buu-Tri Nguyen* and my mother *Huong Dang* have been there for me through thick and thin, so thank you for being there. They have always believed in me that I am capable of doing what I do now.
Special thanks to *Duy Tran* a very good friend of mine with whom I worked with in a virtual office. Although, our fields are different (iOS software development versus research in numerical analysis), it helps a lot to have people surrounding you with a similar kind of motivation. Subsequently, it adds the element of accountability while doing work at home.

Buu-Van Nguyen
Delft, July 2021

Abstract

The single-step single nucleotide polymorphism best linear unbiased prediction (ssSNPBLUP) model can potentially be used in animal breeding for genetic evaluations. It has been reported that this model has convergence issues when the Preconditioned Conjugate Gradient (PCG) method is applied. This is due to the linear system being ill-conditioned. Moreover, in a recent research a subspace decomposition deflation method has been proposed. Unfortunately, the method requires too many deflation vectors, which results in an implementation of the Deflated PCG (DPCG) algorithm that requires a high demand of RAM. In this thesis a different subdomain decomposition strategy is proposed. This subdomain decomposition method utilises the k-means algorithm applied on the matrix of correlation among the single nucleotide polymorphisms (SNPs). This method has been applied on a simulated data set of dairy cattle. This method results in halving the number of deflation vectors required for the same rate of convergence compared with the previous subdomain decomposition deflation method.

In practice for genetic evaluations the data sets increases over time by incorporating new information. In this thesis a specific initial solution has been investigated. This initial solution closely resembles the solution of the updated system to solve the ssSNPBLUP model efficiently. Also a deflation method that utilises the previous solutions of genetic evaluations can be applied. The Proper Orthogonal Decomposition (POD) based deflation method is proposed in this thesis to full fill this requirement. Moreover, only a few deflation vectors are needed to improve the rate of convergence.

Furthermore, the subdomain decomposition deflation method deflates the largest eigenvalues of the preconditioned coefficient matrix and the POD deflation method deflates the smallest eigenvalues of the preconditioned matrix. Combining the deflation vectors of both methods results in an improvement in the rate of convergence compared with the methods on their own.

Based on the results the best performing method is a combination of selecting the initial solution as the previous genetic evaluation, the POD deflation method and the k-means++ clustering applied on the subdomain decomposition deflation method. In this research a reduction of 81% in iteration time and 19% in total computation time has been observed. Note that the initial solution as the previous genetic evaluation approach can only be employed if that solution is available. Likewise, the POD method can only be applied if multiple solutions of genetic evaluations are available.

Summary

It has been reported by Vandenplas et al. in [25] that the single-step single nucleotide polymorphism best linear unbiased prediction (ssSNPBLUP) model has convergence issues when the preconditioned conjugate gradient (PCG) method is applied.
The PCG method can be accelerated by introducing a second-level preconditioner. In this case the subdomain decomposition deflation method is the method of choice, which results in the deflated PCG method (DPCG). The subdomain decomposition method creates subdomains with a fixed size containing single nucleotide polymorphisms (SNPs) that are sampled at random. This method does improve the rate of convergence by affecting the large eigenvalues. Despite this fact it still requires too many deflation vectors to be of practical use. In this thesis a different subdomain decomposition strategy is proposed. This strategy involves the k-means algorithm, which clusters the SNP based on their correlation. This strategy only requires half the deflation vectors to obtain the same rate of convergence compared with the subdomain decomposition strategy studied in [25]. The methodology and results are explained in chapter 7.4.6.
In practice new data is added on a routinely basis. In the thesis this process is referred as updating. The ssSNPBLUP model can incorporate this updated data, but the linear system resulting from it has to be solved again to obtain accurate values of the new animals. An option is to apply the DPCG method, but with the increasing size of the linear system due to the new data, the deflation method becomes more expensive in FLOP count and RAM. One could say to compute less deflation vectors for the subdomain decomposition deflation method, but one could also try an alternative that utilises data from the older system. This data refers to the solutions of the previous solutions.
To accelerate the convergence of the PCG method a specific initial solution has been chosen that closely resembles the solution of the updated system. This initial solution is the solution of the previous genetic evaluation, the method is described in chapter 7.4.3. Additionally, a deflation method can be applied that can utilise this data and only requires a few deflation vectors to improve the rate of convergence. This method is the Proper Orthogonal Decomposition (POD) based deflation method, which is explained in chapter 8.3.2. Moreover, the amount of deflation vectors resulting from the POD method have been reduced from 12 to 1 deflation vector. Also the POD deflation method can be combined with the specific choice of the initial solution to enhance the rate of convergence.
The POD deflation method seems to only affect the smallest eigenvalue and the subdomain decomposition deflation method only affects the largest eigenvalue. Thus a method is proposed to combine the deflation vectors resulting from these methods. This combination now affects both the smallest and largest eigenvalues as effectively as if they are used on their own. This resulted in an improvement in the rate of convergence. The methodology and results are shown in chapter 8.4.8.

Contents

1 Introduction **1**
- 1.1 Research Questions . . . 2
- 1.2 Thesis outline . . . 5

2 Preliminaries **6**
- 2.1 General . . . 6
- 2.2 Eigenvalues and eigenvectors . . . 8
- 2.3 Inner products and norms . . . 9
- 2.4 Probability and Statistic . . . 10

3 Conjugate Gradient Method **13**
- 3.1 Fundamental properties . . . 13
- 3.2 Convergence and Ritz values . . . 16

4 Preconditioned Conjugate Gradient method **20**
- 4.1 Fundamental properties . . . 20
- 4.2 Preconditioner . . . 22

5 Deflated Preconditoned Conjugate Gradient method **23**
- 5.1 DCG . . . 23
- 5.2 DPCG . . . 25
- 5.3 Theory of deflation vectors . . . 27
  - 5.3.1 Eigenvectors . . . 27
  - 5.3.2 Eigenspace . . . 27
- 5.4 Deflation vectors . . . 29

6 Model **31**
- 6.1 Preliminaries . . . 31
- 6.2 Single-trait ssSNPBLUP . . . 33
  - 6.2.1 Model parameters . . . 33
  - 6.2.2 Model derivation . . . 38
- 6.3 Multiple-trait ssSNPBLUP . . . 40
- 6.4 Matrix structure . . . 41
  - 6.4.1 (Co)variance matrix of residual effects R . . . 41
  - 6.4.2 Incidence matrix of fixed effects X . . . 42
  - 6.4.3 Incidence matrix of animal effects W . . . 43

6.4.4 (Co)variance matrix Σ . . . 45
6.5 Subdomain deflation method applied on SNP . . . 45
6.5.1 Subdomain decomposition with random sampling . . . 46

7 Subdomain decomposition strategies **47**
7.1 Data . . . 47
7.2 Julia . . . 47
7.3 Deflation operator applied on the ssSNPBLUP related with ssGBLUP . . . 48
7.4 Subdomain selection strategy . . . 51
7.4.1 Subdomain selection per chromosome . . . 51
7.4.2 Subdomain selection on groups of SNPs within a chromosome . . . 54
7.4.3 Different initial solution with random sampling . . . 56
7.4.4 Map of the chromosome . . . 60
7.4.5 Random grouping of SNPs per subdomain without chromosome restriction . . . 62
7.4.6 k-means clustering . . . 63
7.5 Concluding remarks . . . 79

8 POD Based Deflation **80**
8.1 Data . . . 80
8.2 Estimating EBV for reduced systems . . . 81
8.3 Proper Orthogonal Decomposition (POD) Deflation vectors . . . 87
8.3.1 Singular Value Decomposition . . . 87
8.3.2 Proper Orthogonal Decomposition (POD) . . . 88
8.3.3 Deflation vectors . . . 90
8.3.4 Implementation . . . 91
8.4 Results of the POD deflation method . . . 92
8.4.1 POD 1% case . . . 92
8.4.2 POD with information up to 3% . . . 93
8.4.3 POD method sensitivity of the initial solution . . . 94
8.4.4 Comparison between covariance matrix . . . 96
8.4.5 Removing deflation vectors . . . 98
8.4.6 Removing data . . . 103
8.4.7 Comparison with eigenvector . . . 108
8.4.8 POD method combined with subdomain decomposition . . . 109
8.4.9 POD and k-means++ clustering . . . 112
8.5 Concluding remarks . . . 115

9 Conclusion **117**

10 Appendix **122**
10.1 Derivation PCG . . . 122
10.2 Derivation ssSNPBLUP subdomain decomposition deflation operator . . . 125
10.3 Supplementary k-means algorithm results . . . 128
10.4 Julia set-up and modules . . . 138
10.5 FLOP count of CG, PCG, DCG and DPCG algorithm . . . 142
10.6 Supplementary POD based deflation method results . . . 151

CHAPTER 1

Introduction

Food is an important factor in our daily life. Dairy products are one of the common food items in the west. To consistently produce dairy products, knowledge is needed of different traits e.g. milk yield, protein yield, fat yield, somatic cell counts (for udder diseases), fertility, etc. from the current cows and the dairy cattle progeny. Selecting the proper dams and sires (mother and father, respectively), which results in the desired and improved progeny with optimal performances is of utmost importance. In order to estimate genetic merits, also called breeding values (e.g. milk yield), a prediction model can be used. Predictions, also known as genetic evaluation, can be performed by using data of the animals such as genomic, pedigree and phenotypic information. In general, genomic data includes several thousand single nucleotide polymorphisms (SNPs). Through genetic evaluation the animals can be ranked for selection based on the predicted genetic merits for traits of interest.
Currently, the method of choice is the single-step genomic best linear unbiased prediction (ssGBLUP) [25][22][24]. This model forms a system of linear equations, see chapter 2 for details on system of linear equations. The ssGBLUP includes genomic information by combining genomic and pedigree relationships into a combined genomic-pedigree relationship matrix. This allows for simultaneous modelling of genotyped and non-genotyped animals. The downside of this method is that an inverse of the genomic relationship matrix, G, is required. This matrix can be computed up to approximately 100.000 genotyped animals on current computers [25]. Thus this is a bottleneck, when the data used is larger than the aforementioned number.

In recent research the single-step SNPBLUP (ssSNPBLUP) model has been used for genetic evaluation, because it does not require the inverse of a dense matrix G [25]. This model also has the advantage to estimate SNP effects as random effects. In [22] two ssSNPBLUP models have been studied, the first one is proposed by Mäntysaari and Strandén [14], and the second one by Liu [10]. In this thesis the model proposed by Liu will be studied, see chapter 6 for a derivation of this model. The ssSNPBLUP model yields a sparse linear system of equations with an SPSD (or SPD) coefficient matrix. The preconditioned conjugate gradient (PCG) method is an efficient method to solve linear systems with these properties. Although, convergence issues have been reported by applying a

diagonal or block-diagonal preconditioner [10]. Note that the rate of convergence of the Conjugate Gradient (CG) methods are bounded by a function of the effective condition number of the coefficient matrix. The effective condition numbers depend on the eigenvalues of the coefficient matrix. In [22], a study has been done on the eigenvalues and it has been observed that the largest eigenvalues of the preconditioned system are related to the equations including SNP effects. The deflated preconditioned conjugate gradient (DPCG) method has been used, where the preconditioner is a block-diagonal matrix and the deflation method is based on a subdomain decomposition deflation method. Seemingly, choosing the subdomains in a particular fashion, it eliminated the largest eigenvalues of the preconditioned coefficient matrix. Thereby, the addition of the deflation method improved the convergence. Unfortunately, the DPCG method requires the computation and storage of a dense matrix, which is the Galerkin matrix, see chapter 5 for details on the Galerkin matrix. Moreover, refining the subdomains resulted in a smaller effective condition number. Consequently, more deflation vectors are created, which means that the size of the Galerkin matrix increases. Hence, the DPCG method becomes more computationally expensive.
A new method which applies the k-means algorithm to choose the subdomains based on the data of the SNP effects is presented in chapter 7. This method requires half of the deflation vectors needed to obtain a similar rate of convergence compared with the subdomain deflation method studied in [25].
In genetic evaluations the current data will be updated with new data of animals on a routinely basis. Incorporating this data will change the linear system. Therefore, the linear system has to be solved once again with the DPCG method. However, solutions of the previous genetic evaluations can be used to improve the rate of convergence, see chapter 8. This can be done by choosing an initial solution close to the solution of the updated linear system. A method to construct an approximation of the solution of the updated linear system is described in chapter 7.4.3. Also a POD based deflation method is introduced to construct a few deflation vectors that improve the rate of convergence, see chapter 8.3.2. This POD based deflation method can be used in conjunction with the approximated initial solution method. Additionally, the deflation vectors of the POD based deflation method can be combined with the deflation vectors of the subdomain decomposition method to further improve the rate of convergence, see chapter 8.4.8.

1.1 Research Questions

The ssSNPBLUP model can be used for genetic evaluation, although convergence problems have occurred by using the PCG method. In recent papers from Vandenplas et al. [25][22][24] two different second-level preconditioners approaches have been applied, which are the deflation method and second-level diagonal preconditioner method.
The main goal of this MSc thesis project is to figure out why the chosen second-level preconditioners applied on the PCG method improves the convergence of the coefficient matrix resulting from the ssSNPBLUP model. Hence the main research question is as follows:

Why does the deflation method applied on the preconditioned system of the ssSNPBLUP model improves the convergence?

To be able to research this question rigorously, a fine understanding of the matrix structure is required, see chapter 6. Moreover, the matrix structure plays an important role for the analysis of the convergence of the DPCG method applying subdomain decomposition, see chapter 7.3. The main research questions branches off into two different sub questions. The first question is related to reduce the amount of deflation vector used with the current method seen in [25]. The other question relates to find an efficient strategy for solving a linear system after new data has been incorporated. For the following research sub questions a small data set will be studied.

Subdomain selection strategy

In recent papers from Vandenplas et al. [25][22][24] the deflation method has been applied using subdomain decomposition. The subdomains are divided per trait. Within each trait, the SNP and non-SNP effects are seperated. Subsequently, the SNP effects are decomposed into smaller subdomains. Each subdomain consists of $k \in \mathbb{N}$ SNP effects. The grouping of SNP effects is done by random sampling without replacement. Thus each sampling can yield different deflation vectors, which could possibly affect the rate of convergence. For constructing the subdomains in [25] pseudo-randomness has been used with the same seed. However, one can question if a different seed is used, how that could affect the rate of convergence. It is also possible to choose a different strategy to construct the subdomains. This results in the following research sub question:

Does it matter how subdomains are constructed for the convergence of the DPCG method applied on the ssSNPBLUP model?

Updating

In practice, genetic evaluations are done on a regular routine. From time to time additional phenotype and/or genotype records are provided of new animals. Usually, these are the progeny of the existing animals in the old data. In this thesis the following assumption is made:

The SNP genotype profile and the fixed effect categories (e.g. male/female or farm owner) remain the same over a set period.

In practice, this is reflected through the data updates. This new data can be incorporated in the current coefficient matrix, which results in a larger matrix. This linear system has to be solved to obtain a new solution. The DPCG method is the method of choice. In general, a larger matrix needs more time for computations. However, the computation time of the DPCG method can be reduced by choosing an initial solution close to the solution of the updated system compared with the initial solution as the zero solution. A possibility is choosing the solution from the previous genetic evaluation as an initial solution. The updated linear system resembles the old linear system for a great portion. Thus the updated solution could resemble the old solution. Therefore, the initial solution could be near the solution of the updated system. This means that the CG method will need less iterations to converge. Note that the updated linear system contains more unknowns than the previous systems. Thus a solution of a previous system does not match

the dimensions of the updated system. Hence the solution of the previous genetic evaluation should incorporate the new unknowns to be able to apply it as an initial solution for the updated system. The unknowns correspond to the new animals. Thus estimated breeding values (EBVs) of the new animals are missing. In practice, the missing estimated breeding values for the initial solution are estimated by taking the average of the estimated breeding values of the parents of the animal. The same methodology will be studied in this research.
After a certain amount of updates several solutions have been obtained. These solutions could potentially contain relevant information for the updated system, since these solutions are derived from previous systems that resembles the updated system. To extract this information a Proper Orthogonal Decomposition based deflation method is proposed in this thesis. This methodology has been applied before in a different context than genetic evaluation [6], and it resulted in a good performance with a few deflation vectors. Furthermore, this method can be applied on time dependent problems, which is also remarked in [6]. The updates can be seen as a time like procedure. Henceforth, this is another motivation to apply the POD method in genetic evaluations. To the knowledge of the writer, the POD based deflation method has not been applied before in the context of genetic evaluation. Thus the research sub question is as follows:

How can you accelerate the convergence of the updated system and why does it work?

1.2 Thesis outline

- **Preliminaries:** In chapter 2 the basic theory and definitions used in Linear Algebra will be introduced to support the theory behind the CG methods.

- **Conjugate Gradient method:** Chapter 3 outlines the basic theory related to the Conjugate Gradient method, which should aid as a foundation for the PCG and DPCG method.

- **Preconditioned Conjugate Gradient method:** Chapter 4 covers the preconditioning method, which is a method to improve the convergence of the Conjugate Gradient method. Questions such as why and how this method works are covered.

- **Deflated Preconditioned Conjugate Gradient method:** Chapter 5 involves the deflation method used on the CG method. It is a method that can effectively deal with the unfavourable eigenvectors of the coefficient matrix, such that it leads to an improvement of the convergence rate. Additionally, the deflation method can also be applied on the PCG method and relevant properties will be described.

- **Model:** The system of linear equations that relates to the ssSNPBLUP model will be introduced in chapter 6. This system contains specific block matrices. The properties of these block matrices will be explained. This gives a clear view of the overall matrix.

- **Methodology and Results:** Each sub question has it own dedicated chapter. Chapter 7 focuses on the subdomain selection strategy and chapter 8 focuses on the updating case. Each chapter is structured as follows:
  - The chapter starts with the relevant procedures to create the results.
  - Thereafter, the results are shown and discussed.

- **Conclusion:** Summary of the results and answers on the research questions can be found in chapter 9. Moreover, suggestions for further research are presented here.

CHAPTER 2

Preliminaries

In this research the Conjugate Gradient (CG) method is pivotal for solving the ssS-NPBLUP model. The Deflated Preconditioned Conjugate Gradient (DPCG) method is of our main interest. Therefore, the knowledge of the properties of the CG method is required to have a clear understanding. The properties of the CG method will be explained in chapter 3. Some knowledge of Linear Algebra is needed to be able to understand the properties. In this chapter a brief explanation of relevant Linear Algebra concepts will be given. The properties of the CG method can be extended for the PCG method and DPCG method, which will be described in chapter 5.

2.1 General

The CG method and its variants involve solving linear systems, which are a part of linear algebra. Linear systems are a set of linear equations. A linear equation in the plane of real numbers is defined by:

$$f(x) = ax + d$$

where $a, d \in \mathbb{R}$ are constants and $x \in \mathbb{R}$. A set of $n \in \mathbb{N}$ real linear equations is given by:

$$\begin{aligned}
f_1(x_1, x_2, \ldots, x_n) &= a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n + d_1 \\
f_2(x_1, x_2, \ldots, x_n) &= a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n + d_2 \\
&\vdots \\
f_n(x_1, x_2, \ldots, x_n) &= a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n + d_n
\end{aligned}$$

where $a_{ij} \in \mathbb{R}$ with $i, j \in \{1, 2, \dots, n\}$ and $x_i \in \mathbb{R}$. Using matrix vector notation, define the vectors $x = (x_1, x_2, \dots, x_n)^T$, $f(x) = (f_1(x), f_2(x), \dots, f_n(x))^T$ and the matrix

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \ddots & \dots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}$$

Assume that $f(x) = b$ with $b \in \mathbb{R}^n$ and $d_i = 0 \; \forall i \in \{1, 2, \dots, n\}$, then the linear system can be written as

$$Ax = b \tag{2.1}$$

From now on, in this chapter, the assumption will be made that $A \in \mathbb{R}^{n \times n}$, unless otherwise stated. In this case matrix A is called a square matrix, since the amount of rows and columns are the same.

It is important to know whether a linear system has a solution(s). If the linear system (3.1) has at least one solution, the linear system is called *consistent*. To guarantee a consistent linear system, we have to assume that the right-hand side b is in the range of A which is defined by $\mathcal{R}(A) := \{b \in \mathbb{R}^n : Ax = b \text{ for } x \in \mathbb{R}^n\}$.
Another important notion is linear independence.

Definition 2.1.1. *Define the vectors $v^1, v^2, \dots, v^m \in \mathbb{R}^m \setminus \{0\}$ with $m, n \in \mathbb{N}$. The vectors are independent if and only if*

$$c_1 v^1 + c_2 v^2 + \ldots + c_m v^m = 0$$

for constants $c_1 = c_2 = \ldots = c_m = 0$ where $c_i \in \mathbb{R}$ with $i \in \{1, 2, \dots, m\}$.

Hence the columns of matrix A can only be linearly independent if a column cannot be expressed as a linear combination of the other columns.

Definition 2.1.2. *Matrix A is called a diagonal matrix if all off-diagonal entries equal 0.*

The identity matrix is a diagonal matrix with value 1 on the diagonal. The $n \times n$ dimensional identity matrix is denoted by $I_{n,n}$.

If the columns of matrix A are linearly independent, then an inverse exists.

Definition 2.1.3. *Matrix A is invertible if and only if*

$$A^{-1}A = AA^{-1} = I_{n,n}$$

where A^{-1} is called the inverse of A.

If an inverse exists the matrix is also called nonsigular, otherwise it is a singular matrix. If matrix A is nonsingular, then linear system (3.1) can be rewritten as:

$$x = A^{-1}b$$

Hence a unique solution exists $x \in \mathbb{R}^n \setminus \{0\}$.

Useful properties for theoretical applications are M-matrices and Hessenberg matrices.

Definition 2.1.4. *Matrix A is called an M-matrix if it satisfies the following properties:*

1. $A_{ij} \leq 0$ *for* $i \neq j$ *where* $i, j \in \{1, 2, \ldots, n\}$
2. $A_{ii} > 0$ *for* $i \in \{1, 2 \ldots, n\}$
3. *A is nonsingular*
4. *The entries of A^{-1} are all nonnegative.*

Definition 2.1.5. *Matrix A is called an upper Hessenberg matrix if $A_{ij} = 0$ with $i > j+1$ where $i, j \in \{1, 2, \ldots, n\}$. Matrix A is called a lower Hessenberg matrix if $A_{ij} = 0$ with $j > i + 1$ where $i, j \in \{1, 2, \ldots, n\}$.*

2.2 Eigenvalues and eigenvectors

Eigenvalues are important for analysing the convergence behaviour of the CG method.

Definition 2.2.1. *An eigenvalue λ of a matrix satisfies*

$$Av = \lambda v$$

where $v \in \mathbb{R}^n \setminus \{0\}$ is called the eigenvector.

Definition 2.2.2. *The spectrum of matrix A is denoted by $\sigma(A)$ which is the set of all eigenvalues of A.*

Definition 2.2.3. *The spectral radius of A is denoted by $\rho(A)$ and is defined by*

$$\rho(A) = \max_{i=1,\ldots,n} \{|\lambda_i| : \lambda_i \in \sigma(A)\}$$

The inverse of a matrix only exists for a square matrix and if there is no eigenvalue equal to 0. In practice, it is quite cumbersome to compute the inverse of A. Hence, iterative methods are used like the CG method. The use of the CG method is limited to symmetric positive semi-definite (SPSD) matrices [26]. The matrix properties are defined by the following definitions.

Definition 2.2.4. *The transpose of a matrix $A \in \mathbb{R}^{n \times n}$ with $n \in \mathbb{N}$, is obtained by swapping the columns with rows. The transpose is denoted by A^T.*

Definition 2.2.5. *The matrix A is symmetric if and only if $A = A^T$.*

Definition 2.2.6. *The matrix A is said to be positive semidefinite if and only if $x^T Ax \geq 0$ $\forall x \in \mathbb{R}^n \setminus \{0\}$.*

If strict equality ($>$) holds, then the matrix A is positive definite in the latter definition.
A symmetric matrix has several important properties, that are helpful in analysing the convergence of the CG method.

Theorem 2.2.1. *The eigenvalues of a symmetric matrix are real, see proof in ([26], p.8).*

Definition 2.2.7. *A matrix that satisfies the following property is called an orthogonal matrix*

$$AA^T = A^T A = I_{n\times n} \tag{2.2}$$

In other words the columns of A are mutually orthogonal and normalised, which is also referred as orthonormal.

Theorem 2.2.2. *A symmetric matrix is diagonally orthogonalisable. Thus there exists an orthogonal matrix P and a diagonal matrix D such that $A = P^T DP$, see proof in ([26], p.8).*

2.3 Inner products and norms

The CG algorithm uses inner products. The inner product is also used to define norms, which can be seen as a metric that quantifies distance between two elements.

Definition 2.3.1. *Let $c \in \mathbb{R}$ be a scalar and $x, y, z \in \mathbb{R}^n$ be vectors. The inner product $< \cdot, \cdot >: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ satisfies*

- *symmetry: $< x, y >=< y, x >$*
- *linearity in the first argument: $< c(x + y), z >= c < x, z > +c < y, z >$*
- *positive definite, $< x, x >> 0$ and $< x, x >= 0 \iff x = 0$*

An example of an inner product is the dot product. It is defined by $< x, y >= x^T y = \sum_{i=1}^n x_i y_i$.
If $< x, y >= 0$, $x \neq y$ and $x, y \in \mathbb{R}^n \setminus \{0\}$, then x and y are orthogonal. Also $< x, y >_A= x^T Ay$ is an inner product when A is symmetric positive definite (SPD). Similarly, when $< x, y >_A= 0$, then x and y are called A-orthogonal. These inner products with the former notations will be utilised throughout the thesis.
The theorems that are required for the convergence behaviour of the CG method includes the norm of a vector and matrix.

Definition 2.3.2. *Let $c \in \mathbb{R}$ be a scalar and $x, y \in \mathbb{R}^n$ be vectors. The vector norm $\| \cdot \| : \mathbb{R}^n \to \mathbb{R}$ has these properties*

- *positivity, $\|x\| \geq 0$ and $\|x\| = 0 \iff x = 0$*

- *homogeneity,* $\|cx\| = |c|\|x\|$
- *triangular inequality,* $\|x + y\| \leq \|x\| + \|y\|$

The following norms will be used in this thesis.

- $\|x\|_p = (\sum_{i=1}^{n} |x_i|^p)^{\frac{1}{p}}$ with $1 \leq p < \infty$
- $\|x\|_\infty = \max_{i\in\{1,2,\ldots,n\}} |x_i|$
- $\|x\|_2 = \sqrt{< x, x >}$
- $\|x|_A = \sqrt{< x, x >_A} = \sqrt{< x, Ax >}$

Normalising a vector is taking the vector in the same direction but with its norm equal to 1. When choosing $x \in \mathbb{R}^n \setminus \{0\}$, then the normalised vector is given by $\hat{x} = \frac{x}{\|x\|}$.

The matrix norm has the following form with $A \in \mathbb{R}^{m\times n}$ and $1 \leq p < \infty$

$$\|A\|_p = \sup_{x\in\mathbb{R}\setminus\{0\}} \frac{\|Ax\|_p}{\|x\|_p} = \max_{\|x\|_p=1} \|Ax\|_p$$

Let $\lambda_{min}(A)$ be the smallest non-zero eigenvalue and $\lambda_{max}(A)$ be the largest eigenvalues of matrix A, respectively. If $p = 2$ the matrix norm satisfies $\|A\|_2 = \sqrt{\lambda_{max}(A^TA)}$. If A is SPSD, then $\|A\|_2 = \lambda_{max}(A)$ and $\|A^{-1}\|_2 = \lambda_{min}(A)$. This leads to the following concept, which are condition numbers.

Definition 2.3.3. *Let $1 \leq p < \infty$, the condition number associated to A is defined by*

$$\kappa_p(A) = \|A\|_p\|A^{-1}\|_p$$

Thus for $p = 2$, we obtain:

$$\kappa_2(A) = \|A\|_2\|A^{-1}\|_2 = \frac{\sqrt{\lambda_{max}(A^TA)}}{\sqrt{\lambda_{min}(A^TA)}}$$

If A is SPSD, then the condition number associated to A satisfies the following relation:

$$\kappa_2(A) = \frac{\lambda_{max}(A)}{\lambda_{min}(A)} \tag{2.3}$$

2.4 Probability and Statistic

For the derivation of the system of linear equations from the ssSNPBLUP model in chapter 6, some basic knowledge of probability and statistic definitions is required.

A variable whose value depends on the outcome of a random phenomenon is called a random variable. For example, the outcome of a dice roll can be considered as a random variable.
Define the sample space Ω as the set of all possible outcomes. Define the event space $\mathcal{F}$

which contains all the possible outcome combinations. Define the probability function P which assigns each event in the event space a number between 0 and 1. Thus the sample space of six-sided dice is $\{1, 2, 3, 4, 5, 6\}$. The event space contains all the possible combinations that can be made with the outcomes.

The expected value can be seen as a generalisation of the weighted average. Let X be a random variable defined on a probability space $(\Omega, \mathcal{F}, P)$, then the expected value has the following definition

$$E[X] = \int_{\Omega} X(\omega)\, dP(\omega)$$

where the Lebesgue integral has been used. This definition can be extended for multidimensional random variables. Let the multidimensional random variable in $\mathbb{R}^n$ be:

$$X = \big(X_1, X_2, \ldots, X_n\big)$$

then the expected value is equal to:

$$E[X] = E[\big(X_1, X_2, \ldots, X_n\big)] = \big(E[X_1], E[X_2], \ldots, E[X_n]\big)$$

The expected value has several well-known properties:

1. $E[X + Y] = E[X] + E[Y]$.
2. $E[aX] = aE[X]$ with a being a constant.
3. $E[XY] = E[X]E[Y]$ if X and Y are independent.
4. $E[X] \geq 0$ if $X \geq 0$.
5. $E[X] \leq E[Y]$ if $X \leq Y$.

Let X and Y be random variables. The covariance of X and Y can be defined as:

$$cov(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$$

with $\mu_X = E[X]$ and $\mu_Y = E[Y]$. The covariance has the following properties:

1. $cov(X + Y, Z) = cov(X, Y) + cov(X, Z)$ with random variables X, Y and Z.
2. $cov(aX, Y) = a \cdot cov(X, Y)$ with a being a constant.
3. $cov(X, Y) = cov(Y, X)$
4. If X and Y are independent, then $cov(X, Y) = 0$

When X is a multidimensional random variable, then the (co)variance matrix of X is defined as:

$cov(X, X) = E[(X - \mu_X)(X - \mu_X)^T] =$

$$\begin{bmatrix} cov(X_1, X_1) & cov(X_1, X_2) & \cdots & \cdots & cov(X_1, X_n) \\ cov(X_2, X_1) & cov(X_2, X_2) & cov(X_2, X_3) & \cdots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \cdots & cov(X_{n-1}, X_{n-2}) & cov(X_{n-1}, X_{n-1}) & cov(X_{n-1}, X_n) \\ cov(X_n, X_1) & \cdots & \cdots & cov(X_n, X_{n-1}) & cov(X_n, X_n) \end{bmatrix}$$

Note the (co)variance matrix is symmetric positive semi-definite. Moreover, the (co)variance matrix has the following property. Let A be a matrix of constants which is conformable, then by applying the linearity of the expected value, the following result can be obtained:

$$\begin{aligned} cov(AX, AX) &= E[(AX - A\mu_X)(AX - A\mu_X)^T] \\ &= E[A(X - \mu_X)(X - \mu_X)^T A^T] \\ &= AE[(X - \mu_X)(X - \mu_X)^T]A^T \\ &= Acov(X, X)A^T \end{aligned}$$

The correlation can be defined as:

$$\rho(X, Y) = \frac{cov(X, Y)}{\sigma_X \sigma_Y}$$

The standard deviation σ_X and σ_Y can be obtained by taking the square root of the variance. The variance is defined as:

$$var(X) = cov(X, X)$$

Note that for the correlation the values are between $-1 \leq \rho(X, Y) \leq 1$. A value of 1 means that two random variables are linearly correlated and -1 is given for negative linear correlation.

CHAPTER 3

Conjugate Gradient Method

For this research we are interested in using the Deflated Preconditioned Conjugate Gradient (DPCG) method. It consists of a deflation and a preconditioning matrix applied on a linear system that is solved with the Conjugate Gradient (CG) method. The chapter starts with a brief explanation of basic iterative methods and projection methods, since the CG method is based on those ideas. After these ideas have been presented, the CG method will be examined. It turns out that the convergence of the CG method depends on the spectrum of the linear system. To obtain an improvement in convergence, a more favourable spectrum is needed. This can be done by applying a preconditioning matrix on the linear system of primary interest. Moreover, this method is known as the Preconditioned Conjugate Gradient (PCG) method. This method will be explained in chapter 4. Even after applying a preconditioner, the transformed coefficient matrix could still have an undesirable spectrum. The deflation method can be applied on a preconditioned system to obtain a favourable spectrum. Thus the DPCG method could potentially speed up the convergence. In chapter 5, details about the deflation method will be explained.

3.1 Fundamental properties

The CG method is an iterative method that solves a linear system with specific requirements

$$Ax = b \tag{3.1}$$

with $A \in \mathbb{R}^{n\times n}$ symmetric positive semi-definite (SPSD) matrix, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$ and $n \in \mathbb{N}$. Basic iterative methods are computed by iterating over

$$x^{k+1} = x^k + M^{-1}r^k \tag{3.2}$$

where $x^k \in \mathbb{R}^n$ denotes the k-th approximation and $r^k = b - Ax^k \in \mathbb{R}^n$ the residual of the k-ith iteration, $A = M - N$ with $M, N \in \mathbb{R}^{n\times n}$ and M is nonsingular. Applying the recursion (3.2) for several steps, we have

$$\begin{aligned}
&x^0 \\
&x^1 = x^0 + M^{-1}r^0 \\
&x^2 = x^0 + 2M^{-1}r^0 - M^{-1}AM^{-1}r^0 \\
&\quad\vdots
\end{aligned}$$

Applying this m times, results in

$$x^m \in x^0 + \text{span}\{M^{-1}r^0, M^{-1}A(M^{-1}r^0), \ldots, (M^{-1}A)^m(M^{-1}r^0)\} \tag{3.3}$$

The subspace as seen in (3.3) is known as the Krylov subspace of dimension m corresponding to matrices A, M and the initial residual r^0. It is defined as

$$\mathcal{K}_m(M^{-1}A, M^{-1}r^0) := \text{span}\{M^{-1}r^0, M^{-1}A(M^{-1}r^0), \ldots, (M^{-1}A)^m(M^{-1}r^0)\} \tag{3.4}$$

Hence an approximation of a basic iterative method is an element of $x^0+\mathcal{K}_m(M^{-1}A, M^{-1}r^0)$. For the sake of convenience $\mathcal{K}_m$ will be a shorthand notation for $\mathcal{K}_m(M^{-1}A, M^{-1}r^0)$.

The CG method is also a projection method based on Krylov subspaces [16]. In general a projection method solves the linear system (3.1) by approximating a solution $x^k \in \mathbb{R}^n$ with $k \in \mathbb{N}$ from a subspace $x^0+\mathcal{K}_m$ with x^0 being the initial guess. Additionally, the Petrov-Galerkin condition is imposed on approximating this solution, which is given by

$$b - Ax^k \perp \mathcal{L}_m$$

with $\mathcal{L}_m$ being a subspace of dimension $m \in \mathbb{N}$. The subspace $\mathcal{L}_m$ varies based on the preconditioning and deflation method that is chosen. The approximated solutions that are obtained from a projection method based on Krylov subspaces have the following form

$$A^{-1}b \approx x^k = x^0 + p_{k-1}(A)r^0$$

with p_{k-1} being a matrix polynomial of degree $k-1$, which can be seen from the recursion of an iterative method. If the initial guess is $x^0 = 0$, then from the latter equation, it is easy to see that the matrix polynomial approximates the inverse of A.

The CG method minimises the error with respect to the A-norm

$$\|x - x^k\|_A = \min_{y \in \mathcal{K}_k(A, r^0)} \|x - y\|_A$$

with x the solution of equation (3.1), which leads to the following algorithm:

Algorithm 1: Conjugate Gradient method

```
1  Choose $x^0$;
2  $r^0 = b - Ax^0$;
3  $p^0 = r^0$;
4  for $k = 0, 1, \dots,$ until convergence do
5      $w^k = Ap^k$;
6      $\alpha_k = \frac{<r^k, r^k>}{<p^k, w^k>}$;
7      $x^{k+1} = x^k + \alpha_k p^k$;
8      $r^{k+1} = r^k - \alpha_k w^k$;
9      $\beta_k = \frac{<r^{k+1}, r^{k+1}>}{<r^k, r^k>}$;
10     $p^{k+1} = r^{k+1} + \beta_k p^k$;
```

The algorithm consists of 1 matrix-vector multiplication and 10 vector calculations. Moreover, 1 matrix, 4 vectors and 2 scalars have to be stored in memory. The choices of the stopping criteria will be discussed later in this thesis. Note that $r^{k+1} = b - Ax^{k+1} = b - x^k - \alpha_k Ap^k = r^k - \alpha_k Ap^k$. This avoids one additional matrix-vector multiplication. Due rounding errors, the residual calculated in this way, could significantly deviate from calculating it compared with $r^{k+1} = b - Ax^{k+1}$. Therefore, it is recommended to recompute $b - Ax^{k+1}$ after the stopping criterion has been satisfied [26]. Thus recompute the stopping criterion with $b - Ax^{k+1}$. If it does not satisfy the stopping criterion, then restart the algorithm starting with x^k. Another approach is to compute the residual $b - Ax^{k+1}$ every $m \in \mathbb{N}$ iterations, this is used in applications of genetic evaluations.

The vectors described in algorithm 1 have certain properties conveyed in the following theorem, we assume that $k \in \mathbb{N}$ iterations have been done:

Theorem 3.1.1. *The vectors p^k, r^k and x^k have the following properties*

1. *span*$\{p^0, \dots, p^{k-1}\} =$ *span*$\{r^0, \dots, r^{k-1}\} = \mathcal{K}_k(A; r_0)$
2. $< r^i, r^j >= 0$ *for* $i = 0, \dots, j-1$; $j = 1, \dots, k-1$
3. $< p^i, r^j >= 0$ *for* $i = 1, \dots, j$; $j = 1, \dots, k-1$
4. $< p^i, Ap^j >= 0$ *for* $i = 1, \dots, j-1$; $j = 1, \dots, k-1$
5. $\|x - x^k\|_A = \min_{y \in \mathcal{K}_k(A; r_0)} \|x - y\|_A$

See proof in [7] chapter 10.2.

Some key remarks follow from theorem 3.1.1

- Let $x \in \mathbb{R}^n$ be the solution of (3.1). The Krylov subspace $\mathcal{K}_n(A; r^0)$ related to the CG method forms an orthogonal basis of the vectors $r^0, r^1, \dots, r^{n-1}$. Thus the Krylov subspace is identical to $\mathbb{R}^n$ after n iterations. Moreover, the CG method minimises the error $\|x - y\|_A$ over $K_n(A; r^0)$, consequently the error is equal to 0 after n iterations. Hence the CG method should converge in n iterations. In practice, this does not happen robustly, due rounding errors the properties entailed in theorem 3.1.1 do not hold up. Also in large applications when $n >> 1$, it requires many iterations which could lead up to an enormous computation time.

- The error measured in the A-norm is monotonically decreasing. Since $\mathcal{K}_k(A; r^0) \subset \mathcal{K}_{k+1}(A; r^0)$ [26]. Note that, x, the exact solution is unknown. Therefore, the residual is usually used as a reference for convergence instead of the norm of the error. Although the 2-norm of the residual is not necessarily monotonically decreasing, but it satisfies the following inequality:

$$\|r^k\|_2 = \|b - Ax^k\|_2 = \|A(x - x^k)\|_2 \leq \sqrt{\|A\|_2}\|\|x - x^k\|_A$$

  Note that the error $\|x - x^k\|_A$ is monotonically decreasing, which means that the residual should decrease after several iterations depending on the value of $\|A\|_2$.

- The CG method has granted its name, because the search direction vectors p^k, also known as gradient, are mutually orthogonal in the A-inner product.

- Note that α_k and β_k denominators break down when they equal zero. This only happens when $r^k = 0$, thus the linear system (3.1) is solved. This is also known as a lucky breakdown.

- A proper stopping criterion for the CG method would be

$$\frac{\|r^k\|}{\|b\|} \leq \epsilon \text{ with } \epsilon > 0$$

  This criterion uses the residual r^k after each iterate. The CG method inherently calculates the residual. This criterion is scaling invariant, which means that it yields the same result for any scalar multiple of the linear system. Moreover, a good initial guess x^0 does not increase the amount of iterations needed.

3.2 Convergence and Ritz values

The error of each iterate with the CG method is bounded, which is formulated in the following theorem.

Theorem 3.2.1. *Let x^k be the k-th iterate of the CG method and x the solution. Then the following inequality holds true*

$$\|x - x^k\|_A \leq 2\left(\frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}\right)^k \|x - x^0\|_A$$

See proof in [16] chapter 6.11.3.

From theorem 3.2.1 it can be seen that the method converges quicker if $\kappa_2(A) \approx 1$. The condition number $\kappa_2(A)$ as given by (2.3), tells us that the extreme eigenvalues hold a key part in the convergence of the CG method. Also the upper bound suggests linear convergence behaviour. In practice, this does not hold up. After the initial iterations the method converges faster than the given upper bound with the original condition number. This behaviour is called **super linear convergence**. This convergence behaviour could be caused by a smaller *effective condition number* after several iterations. Moreover, the amount of distinct eigenvalues also influence this behaviour. These convergence behaviour can be seen through a construction of a matrix polynomial. Since the CG method is a Krylov subspace method which minimises the error, the k-th degree matrix polynomial p_k for the k-th iterate can be defined as

$$\|x - x^k\|_A = \|p_k(A)(x - x^0)\|_A$$

where $p_k(0) = 1$ [26]. Note that A is SPSD, thus there exists an orthonormal eigensystem with eigenvalues $0 \le \lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$ and the corresponding orthonormal eigenvectors $v^1, v^2, \ldots, v^n$. The error can be written as

$$x - x^0 = \sum_{i=1}^{n} \gamma_i v^i \tag{3.5}$$

$$x - x^k = \sum_{i=1}^{n} \gamma_i p_k(\lambda_i) v^i \tag{3.6}$$

where $\gamma_i \in \mathbb{R}\ \forall i \in \{1, 2 \ldots, n\}$. If $p_k(\lambda_i) = 0\ \forall i \in \{1, 2, \ldots, n\}$, then $x - x^k = 0$. Additionally, if there are $m < n$ distinct eigenvalues, the error can be rewritten as

$$x - x^k = \sum_{i=1}^{m} \tilde{\gamma}_i p_k(\tilde{\lambda}_i) \tilde{v}^i \tag{3.7}$$

where $\tilde{\lambda}_i$ are all distinct eigenvalues of A, $\tilde{\gamma}_i$ and $\tilde{v}^i$ are adjusted such that the equality still holds. Thus the CG method converges at most within $m < n$ iterations.
Also from equation (3.5) it can be noticed that the smallest and largest eigenvalues of A that influence the convergence, can be defined as

$$a = \min_{i \in \{1,2,\ldots,n\}} \{\lambda_i | \gamma_i \neq 0\}$$

$$b = \max_{i \in \{1,2,\ldots,n\}} \{\lambda_i | \gamma_i \neq 0\}$$

Hence when γ_i equals 0 the error component corresponding to λ_i plays no part in the convergence. Thus the eigenvalue λ_i has converged. From this the *effective condition number* of matrix A can be derived

$$\kappa(A) = \frac{b}{a}$$

Replacing the condition number in theorem 3.2.1 by the effective condition number, a smaller or equal upper bound is obtained.
In practice, the error related to the extreme eigenvalues usually corresponding to the smallest eigenvalue converges first, in other words γ_i equals 0. This means that the effective condition number can be replaced by a smaller condition number, because the denominator is a larger number. Thus from this behaviour superlinear convergence will

occur if the eigenvalues are clustered around a certain point with some outliers.

Knowing the eigenvalues of the coefficient matrix A can be helpful to understand the convergence behaviour. In practice, obtaining eigenvalues can take a lot of computation time. With the CG method the eigenvalues of A can be approximated, whilst solving the linear system $Ax = b$. This can be done, because the CG method can be derived from the Lanczos algorithm which can be derived from the Arnoldi method applied on a SPSD coefficient matrix [16]. The Arnoldi algorithm is a method to obtain an orthogonal basis for a Krylov subspace $\mathcal{K}_m$. As a result from running the Arnoldi algorithm a Hessenberg matrix $\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$ is created.

Theorem 3.2.2. *Denote by $V_m \in \mathbb{R}^{n\times m}$ the matrix with column vectors $v^1, v^2 \dots v^m$ which forms an orthogonal basis for the Krylov subspace $\mathcal{K}_m$. Let $\overline{H}_m \in \mathbb{R}^{(m+1)\times m}$. Define H_m as the matrix obtained from $\overline{H}_m$ by deleting its last row. Then the following relation holds:*

$$V_m^T A V_m = H_m$$

See proof in [16] chapter 6.3.1.

The Lanczos algorithm yields a Hessenberg matrix which is symmetric tridiagonal. This matrix can be used to approximate the smallest and largest eigenvalues of A. This can be useful to approximate the condition number of A. Since the CG method is based on the Lanczos algorithm, it is possible to create a symmetric tridiagonal matrix from the CG method. The diagonal elements are given by:

$$\delta_{k+1} = \begin{cases} \alpha_k^{-1} & \text{for } k = 0 \\ \alpha_k^{-1} + \frac{\beta_{k-1}}{\alpha_{k-1}} & \text{for } k > 0 \end{cases}$$

where α_k and β_k are values obtained from the CG method given in algorithm 1. The co-diagonal elements are obtained by:

$$\eta_{k+1} = \frac{\sqrt{\beta_{k-1}}}{\alpha_{k-1}}$$

This results in the general form of the m-dimensional Lanczos tridiagonal matrix in terms of the CG coefficients:

$$T_m = \begin{bmatrix} \delta_1 & \eta_2 & & & \\ \eta_2 & \delta_2 & \eta_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \cdot & \cdot & \eta_m \\ & & & \eta_m & \delta_m \end{bmatrix}$$

The matrix T_m is also known as the Ritz matrix [26]. The eigenvalues and eigenvectors of the Ritz matrix are called Ritz values and vectors, respectively. These are approximations of eigenvalues and eigenvectors of A. From theorem 3.2.2 the following holds:

$$T_m = R_m^T A R_m$$

where R_m is the matrix that consists of normalised column vectors of the residuals $r^0, r^1, \ldots, r^{m-1}$ which form an orthonormal basis for the Krylov subspace $\mathcal{K}_m$.
The super linear convergence behaviour of the CG method can be explained with Ritz values. When one or more Ritz values have sufficiently converged to the corresponding eigenvalues an improvement in the rate of convergence is expected [20]. This should be equivalent to the notion of an effective condition number and convergence of an eigenvalue as seen in equation (3.7).
In general, Ritz values and vectors are tools used in a theoretical way to describe the convergence behaviour of the CG method [26].

CHAPTER 4

Preconditioned Conjugate Gradient method

In the previous chapter, it has been shown that the CG method convergence relies on the distribution of the spectrum of A. A distribution that is clustered is an ideal situation, which improves the convergence of the CG method. Since the condition number and effective condition number will be close to 1. A method to change the spectrum distribution of A to a more favourable distribution will be introduced in this chapter called preconditioning.

4.1 Fundamental properties

Let $M = LL^T$ be a preconditioner and L be a nonsingular matrix. The system of interest with the preconditioned conjugate gradient method

$$M^{-1}A = M^{-1}b \iff \tilde{A}\tilde{x} = \tilde{b} \tag{4.1}$$

where $\tilde{A} = L^{-1}AL^{-T}$, $\tilde{x} = L^T x$ and $\tilde{b} = L^{-1}b$. Note that M is a symmetric positive definite matrix. The PCG algorithm can be derived by applying the CG algorithm to the linear system (4.1).

Algorithm 2: Preconditioned Conjugate Gradient method

```
1  Choose $x^0$;
2  $r^0 = b - Ax^0$;
3  Solve $My^0 = r^0$;
4  $p^0 = y^0$;
5  for $k = 0, 1, \dots,$ until convergence do
6      $w^k = Ap^k$;
7      $\alpha_k = \frac{<r^k, y^k>}{<p^k, w^k>}$;
8      $x^{k+1} = x^k + \alpha_k p^k$;
9      $r^{k+1} = r^k - \alpha_k w^k$;
10     Solve $My^{k+1} = r^{k+1}$ ;
11     $\beta_k = \frac{<r^{k+1}, y^{k+1}>}{<r^k, y^k>}$;
12     $p^{k+1} = y^{k+1} + \beta_k p^k$;
```

In algorithm 2 we have $z^k = M^{-1}r^k$, this can be calculated without computing the inverse by solving $Mz^k = r^k$. Due to this extra step the PCG algorithm requires an extra solve compared to the CG algorithm. Hence the system involving $M = LL^T$ should be cheap to solve. Otherwise one iteration of the PCG algorithm will become way more expensive than the CG algorithm. In cases where M is a lower triangular matrix, a forward substitution algorithm can be applied.

The PCG method has the following properties

$$x^k \in x^0 + \mathcal{K}^k \left(M^{-1}A; M^{-1}r^0\right) \tag{4.2}$$

$$\|x - x^k\|_A \leq 2 \left(\frac{\sqrt{\kappa_2(M^{-1}A)} - 1}{\sqrt{\kappa_2(M^{-1}A)} + 1} \right)^k \|x - x^0\|_A \tag{4.3}$$

where $r^0 = b - Ax^0$ is the initial residual. The spectrum of the preconditioned system $L^{-1}AL^{-T}$ has the same spectrum as AM^{-1} and $M^{-1}A$. Equivalently, the following equalities hold

$$\sigma(L^{-1}AL^{-T}) = \sigma(AM^{-1}) = \sigma(M^{-1}A)$$

Due to these properties of the PCG method, it is also known as the CG method applied to the preconditioned coefficient matrix $M^{-1}A$ [18]. This can also be useful in acquiring the eigenvectors or eigenvalues, by choosing one of the linear systems that is easier to compute. From equation (4.2) it is clear that the PCG method minimises the error in the A-norm. Taking $P = I$ leads to the CG algorithm. Whilst choosing $L^TL = A$ the method will converge in one iteration, because the upper bound (4.3) equals 0 and the norm is nonnegative. Another reason why the method theoretically converges in one iteration, is by applying $L^TL = A$, the preconditioned system changes to $x = A^{-1}b$.

There are several remarks about the PCG algorithm [26]:

- Similarly as in the CG method, the denominators of α_k and β_k are equal to 0 when the algorithm has converged, because M is assumed to be positive definite.

- The following conjugate properties hold

$$(r^i)^T M^{-1} r^j = 0 \text{ for } i \neq j \tag{4.4}$$
$$(p^i)^T P^{-1} A P^{-T} p^j = 0 \text{ for } i \neq j \tag{4.5}$$

4.2 Preconditioner

A good preconditioner requires three conditions. First, the preconditioner M should be easy to construct. Second, the system $Mz^k = r^k$ should be easy to solve. At last, the eigenvalues of $\tilde{A}$ should be clustered. The latter condition ensures an effective condition number close to 1.

A diagonal scaling can be used such that the diagonal entries of $M^{-1}A$ are equal to 1 or the rows (or columns) are of equal norm. This choice approximately minimises the condition number of $M^{-1}A$ [19][21]. An example of a diagonal scaling, let $M = diag(A)$ be a preconditioner where $diag(A)$ are the diagonal entries of the matrix A. This has the advantage that the CG method can be applied on $M^{-1}Ax = M^{-1}b$ instead of algorithm 2. Additionally, we have $diag(M^{-1}A) = 1$, which results in n less multiplications for the matrix vector product.
This method is readily applied for the ssSNPBLUP model in [25][22][24].

Another preconditioner is the incomplete Cholesky decomposition, which can be applied to any symmetric M-matrix. The preconditioner used with this method equals $M = LL^T$, where L is an incomplete Cholesky decomposition. It satisfies the following equation:

$$\begin{cases} L_{ij} & = 0 \quad \text{if } a_{ij} = 0 \\ (LL^T)_{ij} & = A_{ij} \quad \text{if } a_{ij} = 0 \end{cases}$$

However this method is not ideal for large problems, because the limitation of available RAM. Moreover, the coefficient matrix resulting from the ssSNPBLUP model proposed by Liu is SPSD. Thus the existence of the incomplete Cholesky decomposition for this matrix is not guaranteed.

CHAPTER 5

Deflated Preconditoned Conjugate Gradient method

As previously mentioned in chapter 4, the system of linear equations

$$Ax = b$$

where $A \in \mathbb{R}^{n\times n}$ is an SPSD matrix, and $b \in \mathcal{R}(A)$. The preconditioner M applied on the matrix A results in the matrix $M^{-1}A$. However $M^{-1}A$ still consists of unfavourable eigenvalues. These eigenvalues slow down the convergence of the method. The deflation method can solve this problem efficiently, granting us an improvement in convergence. In this chapter the deflated method will be explained, including a brief overview of the theory behind it. Hereafter, the deflation method combined with the preconditioning will be explained. This chapter will conclude with a showcase of relevant deflation method techniques.

5.1 DCG

Definition 5.1.1. *Let A be an SPSD matrix. Let the deflation-subspace matrix, $Z \in \mathbb{R}^{n\times k}$, have full rank and $k < n - d$, where d is equal to $\dim(\mathcal{N}(A))$. Assume that the Galerkin matrix, $E \in \mathbb{R}^{k\times k}$, is nonsingular, the correction matrix, $Q \in \mathbb{R}^{n\times n}$ and the deflation matrix, $P \in \mathbb{R}^{n\times n}$, are defined in the following way:*

$$P = I - AQ,\; Q = ZE^{-1}Z^T,\; E = Z^TAZ \tag{5.1}$$

The columns of the matrix Z are called *deflation vectors*. The vectors are chosen such that E is a nonsingular matrix. Hence Z is chosen such that $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$. This is shown in the following lemma.

Lemma 5.1.1. *Let A, E and Z be as given in definition 5.1.1. If $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$, then E is nonsingular, see proof ([18], p.26).*

Lemma 5.1.2. *Let A, P and Z be as given in definition 5.1.1. If $\mathcal{N}(A) \not\subseteq \mathcal{R}(Z)$, then we have*

- $PAZ = 0_{n,k}$

- $P^T Z = 0_{n,k}$

See proof in ([18], p.27).

From this lemma it is clear that $\mathcal{R}(Z) \subset \mathcal{N}(PA)$. Note that A is $SPSD$. Moreover, $\mathcal{R}(Z) \cap \mathcal{N}(A) = \emptyset$, since $\mathcal{R}(Z) \nsubseteq \mathcal{N}(A)$, thus we have $\mathcal{N}(PA) = \mathcal{R}(Z) \oplus \mathcal{N}(A)$. Hence P has $k + d$ zero eigenvalues.
The spectrum $\sigma(P)$ only consists of eigenvalues 0 and 1, because P is a projection matrix, $P^2 = P$ [18].

Lemma 5.1.3. *Let A and P be as given in definition 5.1.1. If $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z)$, then PA is SPSD, see proof ([18], p.28).*

Applying the deflation matrix P on a linear system, results in the following linear system:

$$PA\hat{x} = Pb \tag{5.2}$$

with $\hat{x}$ being the solution of the deflated system, since $\mathcal{N}(PA) = \mathcal{R}(Z) \oplus \mathcal{N}(A)$. This linear system can be derived by using the splitting

$$x = (I - P^T)x + P^T x \tag{5.3}$$

where $(I - P^T)x = Qb$. Solving the deflated system only guarantees a solution of this system. However, there is a relation between the solution x of the original system $Ax = b$ and $\hat{x}$ the solution of (5.2), which is shown in the following lemma and corollary.

Lemma 5.1.4. *Let P be as given in 5.1.1 and $\mathcal{R}(Z) \nsubseteq \mathcal{N}(A)$. Let x be the solution of $Ax = b$ and $\hat{x}$ be the solution of (5.2). Then $P^T x = P^T \hat{x}$. See proof in ([18], p.29).*

Corollary 5.1.1. *Let P and Q be as given in definition 5.1.1. Assume that x and $\hat{x}$ are solutions of (3.1) and (5.2), respectively. By rewriting the splitting (5.3), we obtain*

$$\begin{aligned} x &= (I - P^T)x + P^T x \\ \Longleftrightarrow x &= Qb + P^T x \\ \Longleftrightarrow x &= Qb + P^T \hat{x} \end{aligned}$$

See in proof ([18], 29)

Hence, from corollary 5.1.1 to obtain the solution of (3.1), only the deflated linear system (5.2) has to be solved.

PA is a singular matrix, it can only be solved if (5.2) is consistent, thus there should exist a $\hat{x}$ such that $PA\hat{x} = Pb$. In this research we are only interested in feasible solutions, therefore we assume $b \in \mathcal{R}(A)$. This leads to $Pb \in \mathcal{R}(PA)$, since $\mathcal{N}(A) \subset \mathcal{N}(PA)$ and $\mathcal{N}(A) \cap \mathcal{R}(A) = \emptyset$, concluding that (5.2) is consistent. The DCG algorithm is given by:

Algorithm 3: Deflated Conjugate Gradient

```
1  Choose $\hat{x}^0$;
2  $r^0 = b - A\hat{x}^0$;
3  $\hat{r}^0 = Pr^0$;
4  $p^0 = \hat{r}^0$;
5  for $k = 0, 1, \ldots,$ until convergence do
6      $\hat{w}^k = PAp^k$;
7      $\alpha_k = \frac{<\hat{r}^k, \hat{r}^k>}{<p^k, \hat{w}^k>}$;
8      $\hat{x}^{k+1} = \hat{x}^k + \alpha_k p^k$;
9      $\hat{r}^{k+1} = \hat{r}^k - \alpha_k \hat{w}^k$;
10     $\beta_k = \frac{<\hat{r}^{k+1}, \hat{r}^{k+1}>}{<\hat{r}^k, \hat{r}^k>}$;
11     $p^{k+1} = \hat{r}^{k+1} + \beta_k p^k$;
12 $x_{final} = Qb + P^T\hat{x}^{k+1}$;
```

where x_{final} is the last approximation of the solution of (3.1). Likewise, as in equations (4.4) and (4.5), we have similar properties for the DCG method.

- $p^k \notin \mathcal{R}(Z)$, because it is a linear combination of the deflated residuals $\{\hat{r}^k\}$ with $k \in \{0, 1 \ldots, k\}$.
- Let $y \in \mathcal{R}(Z)$, then we have $<\hat{r}^k, y>= 0$.
- Applying the above properties, we can conclude that $<\hat{r}^k, \hat{r}^k>$ and $<\hat{w}^k, p^k>$ are only equal to 0, when the deflated solution $\hat{x}$ has been found.

5.2 DPCG

The deflation method can be applied to a preconditioned system, where the preconditioner M is SPD. This method is called the Deflated Preconditioned Conjugate Gradient method (DPCG). This results in the following linear system:

$$\tilde{P}M^{-\frac{1}{2}}A\hat{x} = \tilde{P}M^{-\frac{1}{2}}b \tag{5.4}$$

with

$$\tilde{P} = I - M^{-\frac{1}{2}}AM^{-\frac{1}{2}}\tilde{Q}, \quad \tilde{Q} = \tilde{Z}\tilde{E}^{-1}\tilde{Z}^T, \quad \tilde{E} = \tilde{Z}^T\tilde{A}\tilde{Z}$$

where $\tilde{Z} \in \mathbb{R}^{n\times k}$ is the preconditioned deflation-subspace matrix. Equation (5.4) can be written in the same form as (5.2) by defining the following equalities:

$$\tilde{A} = M^{-\frac{1}{2}}AM^{-\frac{1}{2}}, \quad \hat{\tilde{x}} = M^{\frac{1}{2}}\hat{x}, \quad \tilde{b} = M^{-\frac{1}{2}}b \tag{5.5}$$

Substituting this into (5.4), we obtain

$$\tilde{P}\tilde{A}\hat{\tilde{x}} = \tilde{P}\tilde{b} \tag{5.6}$$

It easy to see that the the properties described for the DCG method also holds for the DPCG method. The algorithm for the DPCG method is given by:

Algorithm 4: Deflated Preconditioned Conjugate Gradient

```
1  Choose $\hat{x}^0$;
2  $r^0 = b - A\hat{x}^0$;
3  $\hat{r}^0 = Pr^0$;
4  Solve $My^0 = \hat{r}^0$;
5  $p^0 = y^0$;
6  for $k = 0, 1, \ldots,$ until convergence do
7      $\hat{w}^k = PAp^k$;
8      $\alpha_k = \frac{<\hat{r}^k, y^k>}{<p^k, \hat{w}^k>}$;
9      $\hat{x}^{k+1} = \hat{x}^k + \alpha_k p^k$;
10     $\hat{r}^{k+1} = \hat{r}^k - \alpha_k \hat{w}^k$;
11     Solve $My^{k+1} = \hat{r}^{k+1}$;
12     $\beta_k = \frac{<\hat{r}^{k+1}, y^{k+1}>}{<\hat{r}^k, y^k>}$;
13     $p^{k+1} = y^{k+1} + \beta_k p^k$;
14 $x_{final} = Qb + P^T\hat{x}^{k+1}$;
```

Noticeably, the matrix $\tilde{P}$ and $M^{\frac{1}{2}}$ are not involved in algorithm 4. The associated system (5.6) is also denoted as

$$M^{-1}PA\hat{x} = M^{-1}Pb \tag{5.7}$$

Hence a deflation method applied on a preconditioned system results in the same algorithm as applying a preconditioning method on a deflated system. Therefore, the properties that holds for the PCG method, also holds for the DPCG method, since PA can be seen as the coefficient matrix that is SPSD in equation (5.7). There are some important remarks for the DPCG method:

- The following equalities hold for $i \neq j$

$$(\hat{r}^i)^T M^{-1} \hat{r}^j = 0 \tag{5.8}$$
$$(p^i)^T PAp^j = 0 \tag{5.9}$$

  Thus the deflated residual, $\{\hat{r}^k\}$, are orthogonal with respect to M^{-1}. The search directions, $\{p^i\}$, are conjugate with respect to PA.

- The error upper bound of the DPCG is given by

$$\|\hat{x} - \hat{x}^k\|_A \leq 2\|\hat{x} - \hat{x}^0\|_A \left(\frac{\sqrt{\kappa(M^{-1}PA)} - 1}{\sqrt{\kappa(M^{-1}PA)} + 1} \right)^{k+1} \tag{5.10}$$

  Similarly, as the CG and PCG methods, the convergence depends on the effective condition number of A and $M^{-1}A$, respectively. In this case it depends on $\kappa(M^{-1}PA)$.

5.3 Theory of deflation vectors

5.3.1 Eigenvectors

The deflation method can improve the convergence of the CG method by eliminating the unfavourable eigenvalues. This can be done if the columns of the deflation-subspace matrix Z consists of the eigenvectors related to the unfavourable eigenvalues. This is stated in the following theorem:

Theorem 5.3.1. *Let A and P be defined as in definition 5.1.1. Suppose M is a SPD preconditioner. Let $M^{-1}A$ have eigenvalues $\{\lambda_k\}_{k=1}^n$ with corresponding orthonormal eigenvectors $\{v^k\}_{k=1}^n$. If $Z = \begin{bmatrix} v_{d+1} & v_{d+2} & \dots & v_{d+k} \end{bmatrix}$, then the spectrum of $M^{-1}PA$ equals*

$$\sigma(M^{-1}PA) = \{0, \dots, 0, \lambda_{d+k+1}, \dots, \lambda_n\}$$

See proof in ([18], p.33).

Thus from theorem 5.3.1 the unfavourable eigenvalues of $M^{-1}PA$ become 0, when Z contains the eigenvectors corresponding to the unfavourable eigenvalues. From theorem 5.3.1 the following results hold:

Corollary 5.3.1. *Let A, M^{-1} and P be defined as in definition 5.1.1. Let $M^{-1}A$ have eigenvalues $\{\lambda_k\}_{k=1}^n$ with corresponding orthonormal eigenvectors $\{v^k\}_{k=1}^n$. Also let Z be defined as in theorem 5.3.1, then we have*

$$\kappa(M^{-1}PA) \leq \kappa(M^{-1}A)$$

See proof in ([18], p.35).

From corollary 5.3.1, we can conclude that the DPCG method is expected to converge faster than the PCG method. This deflation method is also called "eigenvector deflation" or "spectral deflation".

The eigenvector deflation method has several downsides. The eigenvectors have to be computed, which can be quite expensive to compute in practice. Moreover, the eigenvectors can be dense, which leads to a dense deflation-subspace matrix Z. Henceforth, an ideal deflation method should have none of these downsides, meaning that it should result in a sparse deflation-subspace matrix Z that consists of good approximations of the eigenvectors.
Instead of good approximations of the eigenvectors, the deflation vectors of the deflation subspace matrix Z should approximate the same space as the span of the unfavourable eigenvectors, which will be explained in the next section.

5.3.2 Eigenspace

The following theorems will entail the properties for an *arbitrary deflation-subspace matrix.*

Theorem 5.3.2. *Let A and P be defined as in definition 5.1.1. If Z is full-rank, then*

$$\kappa(PA) \leq \kappa(A)$$

See proof in ([18], p.37).

Theorem 5.3.3. *Let A, P and Z be defined as in definition 5.1.1. Suppose that M is an SPD preconditioner. We have*

$$\kappa(M^{-1}PA) \leq \kappa(M^{-1}A)$$

See proof in ([18], p.38).

Theorem 5.3.2 states that the DCG method is expected to converge faster than the CG method. Likewise, theorem 5.3.3 states that the DPCG method is expected to converge faster than the PCG method. For an improvement in convergence, the span of the columns of the deflation-subspace matrix play an important role, which is shown in the following theorem.

Theorem 5.3.4. *Let Z_1, Z_2 be deflation-subspace matrices and P_1, P_2 be the deflation matrices, respectively. Assume that $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z_1)$ and $\mathcal{N}(A) \nsubseteq \mathcal{R}(Z_2)$. If $\mathcal{R}(Z_1) = \mathcal{R}(Z_2)$, then $M^{-1}P_1A = M^{-1}P_2A$. Also the correction matrices satisfy $Q_1 = Q_2$. See proof in ([18], p.36).*

Hence the deflation matrix P depends on the space that the columns of the deflation-subspace Z span.

Theorem 5.3.5. *Let Z_1, Z_2 and P_1, P_2 be defined as in theorem 5.3.4. Also let A be defined as in definition 5.1.1 and M be an SPD preconditioner. If $\mathcal{R}(Z_1) \subseteq \mathcal{R}(Z_2)$, then $\kappa(M^{-1}P_2A) \leq \kappa(M^{-1}P_1A)$. See proof in ([18], p.36).*

Corollary 5.3.2. *Let A be defined as in definition 5.1.1 and M be an SPD preconditioner. Define the deflation-subspace matrices $Z_i = \begin{bmatrix} z_1 & z_2 & \dots & z_i \end{bmatrix}$ for $i \in \{1, 2, \dots, k\}$. Let P_i be the deflation matrix corresponding to Z_i. Then for the smallest and largest eigenvalues the following inequalities hold*

$$\lambda_{d+2}(M^{-1}P_1A) \leq \lambda_{d+3}(M^{-1}P_2A) \leq \ldots \leq \lambda_{d+k+1}(M^{-1}P_kA)$$

and

$$\lambda_n(M^{-1}P_kA) \leq \lambda_n(M^{-1}P_{k-1}A) \leq \ldots \leq \lambda_n(M^{-1}P_1A)$$

This yields

$$\kappa(M^{-1}P_kA) \leq \kappa(M^{-1}P_{k-1}A) \leq \ldots \leq \kappa(M^{-1}P_1A)$$

See proof in ([18], p.36).

Combining theorem 5.3.5 and corollary 5.3.2, we should expect an improvement in convergence, with an increasing number of columns of Z that are linearly independent of each other. However, if Z is increasing in size, then the DPCG method requires more random access memory (RAM) and more floating point operations. This could lead to an increased computation time for the DPCG method than the PCG method per iteration.

5.4 Deflation vectors

A good deflation method requires 5 conditions [18], which are quite similar to the requirements for the preconditioner. First of all, the deflation-subspace matrix should be sparse. Secondly, the deflation vectors should approximate or be identical to the eigenspace corresponding to the unfavourable eigenvalues. Thirdly, constructing the deflation-subspace matrix Z should be cheap. Fourthly, the method has favourable parallel properties. Last of all, albeit of less importance, the method should be easy to implement in an existing PCG code.

There are several deflation methods and we will describe three of them, namely *approximating eigenvectors*, *recycling information of previous Krylov subspaces*, and *subdomain deflation*. Most of the other deflation methods are related to these approaches [18]. The subdomain method is of our interest, because it has favourable properties that apply for our problem. To give a better overview why the subdomain deflation is the method of choice, the two other methods have to be explained.

Starting off with the approximating eigenvectors method. This method as the name describes approximates the eigenvectors of the system of linear equations where the eigenvectors are the columns of the deflation-subspace matrix. There are several ways to approximate the eigenvectors, which can lead to a sparse or non-sparse deflation-subspace matrix. In practice, the approximation in general requires additional effort and there needs to be sufficient RAM to store the approximated eigenvectors. Moreover, when the system gets larger the method takes more computation time. Hence, to alleviate this problem the number of approximated eigenvectors should be relatively small, especially when the eigenvectors are dense. This in return will decrease the computation time, but the effectiveness of the method decreases, since there are relatively more unfavourable eigenvalues for a larger system in general.

Another method is recycling deflation. Associated approaches are *solution* and *deflation* recycling. The first mentioned method results in a dense deflation-subspace matrix. Also it is not guaranteed that the range of the deflation-subspace method consists of eigenvectors corresponding to the unfavourable eigenvalues. The latter method is based on recycling information of Krylov iterations in GMRES-like methods. Although, this method can also lead to dense deflation vectors. Both these approaches can pose memory difficulties.

The subdomain deflation method is based on decomposing the computational domain, Ω, into k non-overlapping subdomains, Ω_i where $\Omega_i \cap \Omega_j = \emptyset$ with $i, j \in \{1, 2 \ldots, k\}$. For each subdomain there is one or more deflation vectors corresponding to it. From the perspective of a spatial domain, it is straightforward on how to decompose the domain. In practice, the domain does not have to be spatial. For example, in the ssSNPBLUP model the computational domain consists of several type of effects. Within these effects there is a certain categorisation. These layers of sorting can be used as a base on how to decompose the domain. In general, there could be a pattern or clear underlying structure. This could be used to decompose the computational domain into subdomains.
If there is only one deflation vector corresponding to a non-overlapping subdomain, then

the deflation vector can be constructed as follows. Each entry in the deflation vector corresponds to a grid point in the domain. The entry is equal to 1 when the grid point lies in the subdomain. Otherwise, it is equal to 0. By doing so, the deflation-subspace matrix Z has only one non-zero component for each row.
For example, observe a 1-D computational domain Ω which has 6 grid points $\{x_i\}_{i=1}^{6}$ and that is divided into three subdomains, that are $\Omega_1 = \{x_1, x_2, x_3\}$, $\Omega_2 = \{x_4, x_5\}$ and $\Omega_3 = \{x_6\}$. Thus the deflation-subspace matrix has the following form

$$Z = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The subdomain deflation method results in a sparse subspace-deflation matrix and is easy to implement in a current PCG method. However, certain possible issues arise. While Z is sparse the Galerkin matrix E is in general a dense matrix. Moreover, this approach could result in too many deflation vectors, while could result in memory issues or increased computation time.

CHAPTER 6

Model

This chapter will cover the genetic model that is of interest in this thesis, which is the single-step single nucleotide polymorphism best linear unbiased prediction (ssSNPBLUP) model proposed by Liu et al. [10]. The emphasis lies on the system of linear equations that results from the model. Before delving in to the derivation of the ssSNPBLUP model, a paragraph dedicated on the relevant biological terminology will be given, which also includes a list of symbols used in the model. Thereafter, the model will be derived for a single-trait case similar as in [10], but the derivation will be slightly modified such that it results in the same ssSNPBLUP model used in [22] and [24]. In practice, multiple traits are used and in chapter 6.3 a brief explanation on the extension to multiple traits will be given.

6.1 Preliminaries

In this paragraph the reader will be introduced with the relevant biological terminology. This will lead to a better understanding of the model itself.
In selective breeding there are goals in mind for example, an improved milk yield, milk with more protein, less food required for growth or disease prevention and so on. These goals can be achieved by selecting the parents based on their traits. Several definitions are given below, before delving into the reason why selection is based on traits. These definitions can also be found in [1] and [15].

- A **phenotype** is the observable and measurable trait of an organism.
- A **genotype** is the genetic make-up of an organism and contains genes.
- A **gene** is a sequence DNA (or RNA), which describes the cellular functions of an organism. Genes are also inheritable, thus they can be passes down to the progeny.
- **Additive genes** are the collection of genes that affect the same trait.
- A **genome** is defined as all genetic material of an organism. Thus including all the DNA of the organism.

- **DNA** also known as deoxyribonucleic acid is a double-stranded helix structure of molecules that carries the genetic information. The DNA is structured through nucleotides.
- A **nucleotide** is a molecule structure which is a building block for DNA. A nucleotide can have one of the following bases: adenine (A), cytosine (C), guanine (G) and thymine (T).
- A **chromosome** holds the DNA. An organism usually has a specific set numbers of chromosomes, which explains the genetic makeup of an organism.
- A **locus** is the position of a gene or series of nucleotides on a chromosome. The plural form is called **loci**.
- **Unlinked loci** are a set of loci that are on different chromosomes or are far apart from each other on the same chromosome.

In general, the following assumption is taken for the relation between traits and genes:

Traits are determined by infinitely many additive genes of infinitesimal effects at unlinked loci.

In other words, there are infinite many genes that contribute towards the expression of a trait. Since genes can be passed down, traits are chosen as a selection criteria.

The genotype, phenotype and pedigree information are used for modelling the relationships among animals. This information of a specific organism can be used in a genetic model to make predictions of the genetic potential of the same kind of organism. The values predicted are also called estimated breeding values (EBV). A straightforward use of a genetic model is ranking animals based on their EBV, then selecting the proper animals for breeding based on the EBV. Predicting the genetic potential has its advantages. For example, the milk yield of a cow can only be measured in a female cow and only after it has given its first birth. Thus the information pertaining to milk yield is restricted and can only be measured when the cow has reached a certain stage in the life cycle. Predicting this information can be done beforehand by predicting it with a genetic model. Although, there is a certain accuracy for these predictions.

The ssSNPBLUP model uses information relevant to the animal. Thus pedigree, genomic and phenotypic information of genotyped and nongenotyped animals are analysed simultaneously. The pedigree information is used, since traits of animals are inheritable. Thus animals that are closely related to one and another share common information e.g., mother, father, sisters, brother and so on. The phenotypic information is valuable, since those are the traits that can be observed and measured of the relevant animals. The genomic information is the genetic makeup of an organism, which is contained in the chromosomes of an animal. Small variations in the genes can lead to different development of traits of an animal. The variations in the genes are called **alleles**. The changes can be observed in the nucleotides, because genes are part of the DNA and DNA is build on nucleotides. If a single nucleotide is different in a DNA sequence, then that difference is called a **single nucleotide polymorphism** (SNP). These variations can be beneficial or disadvantageous and can be inherited as well. Moreover, the SNPs can be mapped by using SNP markers with a chip. The animals with available genomic information are called **genotyped** animals. If the information is not available, then the animals are called **nongenotyped** animals. Note that the ssSNPBLUP model estimates breeding values. Breeding values can be defined as the average additive effects of genes an individual receives from both parents, see [13]. In practice, the effects of all genes cannot be estimated, because they are simply not known. A set of SNPs are used which are assumed to be linked to the unknown genes. Thus the SNP effects give us an estimation of the unknown genes. However, the SNPs are not perfectly linked with the unknown genes. Meaning that the SNPs do not contain all the information of a gene. In other words, the sum of the SNP effects is not equal to the sum of gene effects on the phenotypes. The difference between these two values is called the **residual polygenic effect**.

In the research papers of Vandenplas et al. [25][22][24], datasets of cows have been used. Cows are diploids, meaning that the chromosomes come in pairs. The pairs are usually copies of each other. Although, variations in the gene on a specific location on the chromosome (locus) can occur in a pair. If the pairs are different from one and another it is also called a **heterozygous** gene. If they are the same, then it is called a **homozygous** gene.

A list of symbols and variables defined for the ssSNPBLUP model are shown in table 6.1.

Table 6.1: List of symbols

Symbol	**Description**
m	amount of fitted SNP markers
n	number of animals with $n = n_g + n_n$
n_g	number genotyped animals
n_n	number of nongenotyped animals
n_F	number of fixed effects
n_T	number of phenotype records with $n_T = n_{T,g} + n_{T,n}$
$n_{T,g}$	number of phenotypic records corresponding to genotyped animals
$n_{T,n}$	number of phenotypic records corresponding to nongenotyped animals

6.2 Single-trait ssSNPBLUP

6.2.1 Model parameters

The mixed linear model proposed by Liu has the following form:

$$y = X\beta + \begin{bmatrix} W_n & 0 \\ 0 & W_g \end{bmatrix} \begin{bmatrix} u_n \\ u_g \end{bmatrix} + e \tag{6.1}$$

where the variables are defined as:

- $\beta \in \mathbb{R}^{n_F}$ is a vector of all fixed effects
- $u_g \in \mathbb{R}^{n_g}$ is a vector of additive genetic effects of genotyped animals.
- $u_n \in \mathbb{R}^{n_n}$ is a vector of additive genetic effects of nongenotyped animals.
- $y \in \mathbb{R}^{n_T}$ is a vector of phenotypic records.
- $W_g \in \mathbb{R}^{n_{T,g} \times n_g}$ is a matrix relating records to the corresponding additive genetic effects of the genotyped animals.
- $W_n \in \mathbb{R}^{n_{T,n} \times n_n}$ is a matrix relating records to the corresponding additive genetic effects of the nongenotyped animals.
- $X \in \mathbb{R}^{n_T \times n_F}$ is a matrix relating records corresponding to the fixed effects.

The ssSNPBLUP model used in [22] and [24] does not include the nongenetic random effects, unlike the ssSNPBLUP model described in [10]. In practice, the number of fixed and nongenetic random effects is trait dependent. Thus it can vary per system of linear equations.

Assume that y and u follow a multivariate normal distribution with the following parameters:

$$\begin{aligned} y &\sim \mathcal{N}(X\beta + Wu, R) \\ u &\sim \mathcal{N}(0, var(u)) \end{aligned}$$

where $u = \begin{bmatrix} u_n & u_g \end{bmatrix}^T$ and $R = var(e) \in \mathbb{R}^{n_T \times n_T}$ is the (co)variance matrix of the residual effect, which is defined as:

$$R = \begin{bmatrix} R_n & 0 \\ 0 & R_g \end{bmatrix}$$

where $R_g \in \mathbb{R}^{n_{T,g} \times n_{T,g}}$ and $R_n \in \mathbb{R}^{n_{T,n} \times n_{T,n}}$. The inverse of R can be defined as:

$$R^{-1} = \begin{bmatrix} R_n^{-1} & 0 \\ 0 & R_g^{-1} \end{bmatrix}$$

where R_n^{-1} and R_g^{-1} denotes the inverse of R_n and R_g, respectively. The joint probability density function $f(y, u)$ can now be formulated as:

$$f(y, u) = f_y(y|u) f_u(u) \tag{6.2}$$

The probability density functions are given by:

$$f_y(y|u) = (2\pi)^{-\frac{n_T}{2}} \det(R)^{-\frac{1}{2}} e^{-\frac{1}{2}(y - X\beta - Wu)^T R^{-1} (y - X\beta - Wu)} \tag{6.3}$$

$$f_u(u) = (2\pi)^{-\frac{n}{2}} \det(var(u))^{-\frac{1}{2}} e^{-\frac{1}{2} u^T var(u)^{-1} u} \tag{6.4}$$

The derivatives of the log likelihood are given by:

$$\frac{\partial \log(f(y,u))}{\partial \beta} = -X^T R^{-1}(y - X\beta - Wu) \tag{6.5}$$

$$\frac{\partial \log(f(y,u))}{\partial u} = -W^T R^{-1}(y - X\beta - Wu) - var(u)^{-1}u \tag{6.6}$$

The (co)variance matrix of additive genetics effects, $var(u)$, will be derived later on in this chapter, where the **additive genetic effect** can be described as the effect of the gene.

Assume that there are m SNP markers chosen from a SNP chip for genomic evaluation that cannot explain all ***additive genetic variance****,* $\boldsymbol{\sigma_u^2}$. This results in a proportion $w \in (0,1)$ of additive genetic variance that is not explained by all SNP markers. This proportion is the **residual polygenic variance**. Moreover, additive genetic effects of the genotyped animals can be split into two components [10]:

$$u_g = Zg + a_g \tag{6.7}$$

The variables are defined as:

- $a_g \in \mathbb{R}^{n_g}$ is a vector of residual polygenic effects (RPG).
- $g \in \mathbb{R}^m$ is a vector of additive genetic effects of the m fitted SNP markers.
- $u_g \in \mathbb{R}^{n_g}$ is vector of additive effects of genotyped animals.
- $Z \in \mathbb{R}^{n_g \times m}$ is a design matrix containing the SNP genotypes centred by 2 times the allele frequencies corresponding to the SNP genotype. The matrix is usually defined as $Z = M - 2p$, where p is a vector that contains the observed allele frequencies of all SNPs and M is the SNP genotype matrix with its entries equal to 0 for one homozygous genotype, 1 for the heterozygous genotype, and 2 for the alternate homozygous genotype. Note that the vector $2p$ is equal to the mean of each column of M.

  An example of calculating the allele frequency will be given. Let there be 100 genotyped animals and let there be two allele variants A and a. There are 3 possible genotypes for the animals, which are AA, aa and Aa. The first two genotypes are homozygous and the last one is heterozygous. Let 42 genotyped animals have the AA variant, 18 genotyped animals have the aa variant and 40 genotyped animals have the Aa variant. The allele frequency of A can be calculated with the following formula:

$$p_A = \frac{\#AA + \frac{1}{2}\#Aa}{n_g} = \frac{42 + \frac{1}{2} \cdot 40}{100} = 0.62$$

  Analogously, the allele frequency of a can be calculated. It can also be derived from the following relation:

$$p_A + p_a = 1 \iff p_a = 1 - p_A = 0.38$$

Let the (co)variance matrix of SNP marker effects be defined as:

$$var(g) = \sigma_u^2 B$$

Assume that the SNP effects are uncorrelated and all m SNP markers explain equal additive genetic variance, then

$$\begin{aligned} B &= \frac{1-w}{k} I_{m,m} \\ var(g) &= \sigma_u^2 \frac{1-w}{k} I_{m,m} \end{aligned} \tag{6.8}$$

where $k = 2\sum p_i(1-p_i)$ with p_i being the allele frequency of the i-th SNP.

The pedigree relationship between nongenotyped animals and genotyped animals are included in the following matrix:

$$A = \begin{bmatrix} A_{nn} & A_{ng} \\ A_{gn} & A_{gg} \end{bmatrix}$$

where the block-matrix $A_{gg} \in \mathbb{R}^{n_g \times n_g}$ represents the pedigree relationship between genotyped animals, $A_{nn} \in \mathbb{R}^{n_n \times n_n}$ represents the pedigree relationship between nongenotyped animals, and $A_{gn} \in \mathbb{R}^{n_g \times n_n}$ the pedigree relationship between genotyped and nongenotyped animals. By construction of the pedigree relationship matrix is symmetric. The inverse of the pedigree relationship matrix is defined as:

$$A^{-1} = \begin{bmatrix} A^{nn} & A^{ng} \\ A^{gn} & A^{gg} \end{bmatrix}$$

The variance of the RPG is given by:

$$var(a_g) = w\sigma_u^2 A_{gg} \tag{6.9}$$

Assume that there is no correlation between the SNP marker effects and RPG, then the following relation hold:

$$Cov(a_g, g) = 0 \tag{6.10}$$

Applying equation (6.7), (6.8), (6.9) and (6.10), then the (co)variance matrix of the additive genetic effect of genotyped animals can be denoted as:

$$\begin{aligned} var(u_g) &\overset{(6.7)}{=} var(Zg + a_g) \\ &= cov(Zg + a_g, Zg + a_g) \\ &= cov(Zg, Zg) + cov(Zg, a_g) + cov(a_g, Zg) + cov(a_g, a_g) \\ &\overset{lin.}{=} Zvar(g)Z^T + Zcov(g, a_g) + cov(a_g, g)Z^T + var(a_g) \\ &\overset{(6.8)(6.9)(6.10)}{=} \sigma_u^2 ZBZ^T + Z \cdot 0 + 0 \cdot Z^T + w\sigma_u^2 A_{gg} \\ &= \sigma_u^2(ZBZ^T + wA_{gg}) \end{aligned} \tag{6.11}$$

Let the genomic relationship matrix be defined as:

$$G_{gg} = ZBZ^T + wA_{gg}$$

then substituting this expression in equation (6.11) results in:

$$var(u_g) = G_{gg}\sigma_u^2 \tag{6.12}$$

For the definition of the additive genetic effect of nongenotyped animals, the following matrix is required:

$$T = A_{ng}A_{gg}^{-1}$$

and define d as the vector of deviation effects with the following (co)variance matrix:

$$var(d) = D\sigma_u^2 \tag{6.13}$$

where $D = (A^{nn})^{-1}$. The additive genetic effect of nongenotyped animals can be denoted as:

$$u_n = Tu_g + d \tag{6.14}$$

see [10] for details. *Moreover, assume that there is zero covariance between the additive genetic effects of genotyped animals u_g and deviation effects d, which means that the following holds*:

$$cov(u_g, d) = 0 \tag{6.15}$$

The (co)variance matrix of additive genetic effects of nongenotyped animals is given by:

$$\begin{aligned}
var(u_n) &\overset{(6.14)}{=} var(Tu_g + d)\\
&= cov(Tu_g + d, Tu_g + d)\\
&= cov(Tu_g, Tu_g) + cov(Tu_g, d) + cov(d, Tu_g) + cov(d, d)\\
&\overset{lin.}{=} var(Tu_g) + Tcov(u_g, d) + cov(d, u_g)T^T + var(d)\\
&\overset{lin.\text{ and }(6.15)}{=} Tvar(u_g)T^T + T \cdot 0 + 0 \cdot T + var(d)\\
&\overset{(6.12)(6.13)}{=} \left(TG_{gg}T^T + D\right)\sigma_u^2
\end{aligned} \tag{6.16}$$

The (co)variance matrix of additive genetic effects between genotyped and nongenotyped animals is given by:

$$\begin{aligned}
cov(u_n, u_g) &\overset{(6.14)}{=} cov(Tu_g + d, u_g)\\
&\overset{lin.}{=} cov(Tu_g, u_g) + cov(d, u_g)\\
&\overset{lin.\text{ and }(6.15)}{=} Tcov(u_g, u_g) + 0\\
&\overset{(6.12)}{=} TG_{gg}\sigma_u^2
\end{aligned} \tag{6.17}$$

By definition of the (co)variance matrix:

$$cov(u_g, u_n) = cov(u_n, u_g)^T \tag{6.18}$$

Define the matrix G as:

$$G = \begin{bmatrix} TG_{gg}T^T & TG_{gg} \\ G_{gg}T^T & G_{gg} \end{bmatrix} \tag{6.19}$$

By combining the results from equation (6.12), (6.16), (6.17) and (6.18), the (co)variance matrix of additive genetic effects of animals can be defined as:

$$var(u) = var \begin{pmatrix} u_n \\ u_g \end{pmatrix} = \begin{bmatrix} var(u_n) & cov(u_n, u_g) \\ cov(u_g, u_n) & var(u_g) \end{bmatrix} \overset{(6.19)}{=} \sigma_u^2 G \tag{6.20}$$

where the inverse of matrix G has the following form [5]:

$$G^{-1} = A^{-1} + \begin{bmatrix} 0 & 0 \\ 0 & G_{gg}^{-1} - A_{gg}^{-1} \end{bmatrix} \tag{6.21}$$

The derivation of the inverse of G can be found in (Appendix A, [10]).

6.2.2 Model derivation

Deriving the maximum likelihood estimator (MLE) from the log likelihood $\log(f(y,u))$ for β and u and rearranging the equations yields the Henderson's mixed model equations (MME) [8]. Since R and $var(u)$ are positive definite by definition, the (local) maximum can be obtained by setting the derivatives of the log likelihood (6.5) and (6.6) to 0. This results in the following linear system:

$$\begin{bmatrix} X^TR^{-1}X & X^TR^{-1}W \\ W^TR^{-1}X & W^TR^{-1}W + var(u)^{-1} \end{bmatrix} \begin{bmatrix} \hat{\beta} \\ \hat{u} \end{bmatrix} = \begin{bmatrix} X^TR^{-1}y \\ W^TR^{-1}y \end{bmatrix} \tag{6.22}$$

where $\hat{\beta}$ and $\hat{u}$ are estimators of fixed effects and additive genetic effects. Substituting (6.20) into the linear system (6.22) and replacing G^{-1} with (6.21), we obtain:

$$C^* = \begin{bmatrix} X^TR^{-1}X & X_n^TR_n^{-1}W_n & X_g^TR_g^{-1}W_g \\ W_n^TR_n^{-1}X_n & W_n^TR_n^{-1}W_n + \sigma_u^{-2}A^{nn} & \sigma_u^{-2}A^{ng} \\ W_g^TR_g^{-1}X_g & \sigma_u^{-2}A^{gn} & W_g^TR_g^{-1}W_g + \sigma_u^{-2}\left(A^{gg} + G_{gg}^{-1} - A_{gg}^{-1}\right) \end{bmatrix} \tag{6.23}$$

$$x^* = \begin{bmatrix} \hat{\beta} \\ \hat{u}_n \\ \hat{u}_g \end{bmatrix}, \quad b^* = \begin{bmatrix} X^TR^{-1}y \\ W_n^TR_n^{-1}y_n \\ W_g^TR_g^{-1}y_g \end{bmatrix} \tag{6.24}$$

where the variables are defined as:

- $y_g \in \mathbb{R}^{n_{T,g}}$ phenotypic records corresponding to genotyped
- $y_g \in \mathbb{R}^{n_{T,n}}$ phenotypic records corresponding to nongenotyped
- $X_g \in \mathbb{R}^{n_{T,g} \times n_F}$ a matrix corresponding to the fixed effects of nongenotyped animals
- $X_n \in \mathbb{R}^{n_{T,n} \times n_F}$ a matrix corresponding to the fixed effects of genotyped animals

ssSNPBLUP involves estimations of SNP effects. This can be done by appending, g, the vector of SNP markers effects to u. Define the new vector as:

$$h = \begin{pmatrix} u_n & u_g & g \end{pmatrix}^T$$

which has the following (co)variance matrix with dimension $(m+n) \times (m+n)$:

$$var(h) = var \begin{pmatrix} u_n \\ u_g \\ g \end{pmatrix} = \Sigma$$

Likewise, the inverse of the (co)variance matrix associated with h has the following form:

$$\begin{aligned} \Sigma^{-1} &= \begin{bmatrix} \Sigma^{11} & \Sigma^{12} & \Sigma^{13} \\ \Sigma^{21} & \Sigma^{22} & \Sigma^{23} \\ \Sigma^{31} & \Sigma^{32} & \Sigma^{33} \end{bmatrix} \\ &= \sigma_u^{-2} \begin{bmatrix} A^{nn} & A^{ng} & 0 \\ A^{gn} & A^{gg} + \left(\frac{1}{w} - 1\right) A_{gg}^{-1} & -\frac{1}{w} A_{gg}^{-1} Z \\ 0 & -\frac{1}{w} Z^T A_{gg}^{-1} & B^{-1} + \frac{1}{w} Z^T A_{gg}^{-1} Z \end{bmatrix} \end{aligned} \tag{6.25}$$

see ([10], Appendix A) for details on the derivation. Replacing u with h with the current model, grants this model:

$$\begin{aligned} y &\sim \mathcal{N}(X\beta + \begin{bmatrix} W & 0 \end{bmatrix} h, R) \\ h &\sim \mathcal{N}(0, var(h)) \end{aligned}$$

Analogously, following the derivation with u to obtain the MLE, yields the following linear system with h:

$$C = \begin{bmatrix} X^T R^{-1} X & X_n^T R_n^{-1} W_n & X_g^T R_g^{-1} W_g & 0 \\ W_n^T R_n^{-1} X_n & W_n^T R_n^{-1} W_n + \Sigma^{11} & \Sigma^{12} & \Sigma^{13} \\ W_g^T R_g^{-1} X_g & \Sigma^{21} & W_g^T R_g^{-1} W_g + \Sigma^{22} & \Sigma^{23} \\ 0 & \Sigma^{31} & \Sigma^{32} & \Sigma^{33} \end{bmatrix} \tag{6.26}$$

$$x = \begin{bmatrix} \hat{\beta} \\ \hat{u}_n \\ \hat{u}_g \\ \hat{g} \end{bmatrix}, \quad b = \begin{bmatrix} X^T R^{-1} y \\ W_n^T R_n^{-1} y_n \\ W_g^T R_g^{-1} y_g \\ 0 \end{bmatrix} \tag{6.27}$$

Thus obtaining the following system of linear equations:

$$Cx = b$$

The coefficient matrix C given in (6.26) is a symmetric matrix, since the (co)variance matrices R and Σ are SPD by definition. The block matrix $X^T R^{-1} X$ might be SPSD. This can only occur when more than 1 fixed effect is used. As a consequence C might be SPSD.

6.3 Multiple-trait ssSNPBLUP

The ssSNPBLUP model in the previous section only includes a single trait. In practice, multiple traits are used [10]. The model proposed by Liu et al. can be extended to multiple traits. Let T be the total amount of traits and $t, t' \in \{1, 2, \dots, T\}$. The (co)variance matrix among traits is given by:

$$G_0 = SQS^T \tag{6.28}$$

where S is a matrix of the additive genetic standard deviations and Q is a genetic correlation matrix among traits. Both matrices are symmetric with dimensions $T \times T$ defined as:

$$S = \begin{bmatrix} \sigma_{u_1} & 0 & \cdots & 0 \\ 0 & \sigma_{u_2} & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_{u_T} \end{bmatrix}, \; Q = \begin{bmatrix} 1 & q_{1,2} & q_{1,3} & \cdots & q_{1,T} \\ q_{2,1} & 1 & q_{2,3} & \cdots & q_{2,T} \\ q_{3,1} & q_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & q_{T-1,T} \\ q_{T,1} & q_{T,2} & \cdots & q_{T,T-1} & 1 \end{bmatrix}$$

with σ_{u_t} being the additive standard deviation of trait t and $q_{t,t'}$ the genetic correlation between traits t and t'. By ordering the SNP effects by traits nested within a SNP marker, the (co)variance matrix of the SNP marker effects can be written as:

$$var(g) = \begin{bmatrix} SQ_1S^T & & & 0 \\ & SQ_2S^T & & \\ & & \ddots & \\ 0 & & & SQ_mS^T \end{bmatrix} \tag{6.29}$$

where Q_j is the correlation of SNP effects between traits, given as

$$Q_j = \frac{1-w}{k} \begin{bmatrix} 1 & q_{1,2} & q_{1,3} & \cdots & q_{1,T} \\ q_{2,1} & 1 & q_{2,3} & \cdots & q_{2,T} \\ q_{3,1} & q_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & q_{T-1,T} \\ q_{T,1} & q_{T,2} & \cdots & q_{T,T-1} & 1 \end{bmatrix} \tag{6.30}$$

with j denoting j-th SNP marker with $j \in \{1, 2, \dots, m\}$. In equation (6.30) it is assumed that the correlation of SNP effects between traits to be equal to the genetic correlation between traits (6.28). This assumption only holds if the same residual polygenic variance parameter w is assumed for all traits in the multiple-trait ssSNPBLUP model [10]. Note, that this model assumes equal SNP variances (6.8). The matrix shown in (6.29) can be rewritten as:

$$var(g) = \frac{1-w}{k} I_{m,m} \otimes G_0$$

The design matrix Z containing the SNP genotypes also changes accordingly for a multi-trait model. The matrix Z has dimension $n_gT \times mT$ with n_g the amount of genotyped animals. Ordering the SNP effects by traits within m SNP markers, we obtain:

$$Z = \begin{bmatrix} z_{1,1} & z_{1,2} & \cdots & \cdots & \cdots & z_{1,m} \\ z_{2,1} & z_{2,2} & z_{2,3} & \cdots & \cdots & z_{2,m} \\ \vdots & \ddots & \ddots & \ddots & \cdots & \vdots \\ \vdots & \cdots & \cdots & z_{n_g-1,m-2} & z_{n_g-1,m-1} & z_{n_g-1,m} \\ z_{n_g,1} & \cdots & \cdots & \cdots & z_{n_g,m-1} & z_{n_g,m} \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

where the matrix of only ones has dimensions $T \times T$.

In [24] the single-trait model can be extended to the multi-trait model by replacing equation (6.25) with:

$$G_0^{-1} \otimes \begin{bmatrix} A^{nn} & A^{ng} & 0 \\ A^{gn} & A^{gg} + \left(\frac{1}{w} - 1\right) A_{gg}^{-1} & -\frac{1}{w} A_{gg}^{-1} Z \\ 0 & -\frac{1}{w} Z^T A_{gg}^{-1} & \frac{m}{1-w} I + \frac{1}{w} Z^T A_{gg}^{-1} Z \end{bmatrix} \tag{6.31}$$

This results in the same system of linear equations shown in [22]. Note that G_0^{-1} is on the left side of the Kronecker product, since the animals are sorted within traits. The other way around would result in a right side multiplication.

6.4 Matrix structure

In this chapter the structure of several block-matrices contained in the single-trait ssSNPBLUP model proposed by Liu et al. will be explained. These block-matrices are the incidence matrix of fixed effects X, the incidence matrix of animal effects W and the (co)variance matrix of h which is Σ. The latter block-matrix contains several other block-matrices, including the inverse of the pedigree matrix A.

6.4.1 (Co)variance matrix of residual effects R

In practice, the (co)variance matrix $R \in \mathbb{R}^{n_T \times n_T}$ is usually a diagonal matrix. When a single trait is used the matrix can be defined as:

$$I_{n_T,n_T} \cdot var_e$$

where var_e is the variance of the residual effect. This can be generalised for multiple-traits. Let there be $T \geq 1$ amount of different traits. Then the (co)variance matrix of residual effects is obtained by:

$$I_{n_T,n_T} \otimes var_e$$

where var_e is a diagonal matrix with each diagonal entry corresponding to the variance of the residual effect of a trait.

6.4.2 Incidence matrix of fixed effects X

The first block-matrix entry of the ssSNPBLUP model proposed by Liu et al. is given by:

$$X^T R^{-1} X$$

where $X \in \mathbb{R}^{n\times n}$ is the incidence matrix of fixed effects and $R \in \mathbb{R}^{n\times n}$ is the (co)variance matrix of the residual effects. The row size is equal to the amount of animals used in the model. The column size depends on the fixed effects and their corresponding levels. If there are no fixed effects, then X should be empty. However, it is common practice to construct X with one column with its entries set to value 1. This should represent the overall mean of the model. This is common practice for genetic evaluations to obtain an accurate model without other fixed effects. Moreover, this will count as a fixed effect with one level. In practice, this column is omitted if there is at least 1 fixed effect. A small example of constructing the incidence matrix X from table 6.2 will be given.

Table 6.2: The gender of a cow used as a fixed effect which has 2 levels as well as the ownership of the cow is used as fixed with 2 levels.

Cow	Male	Female	Farmer A	Farmer B
1	1	0	1	0
2	1	0	1	0
3	0	1	1	0
4	0	1	1	0
5	1	0	1	0
6	0	1	0	1

Thus from table 6.2, the following incidence matrix with 2 fixed effects can be obtained:

$$X = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Note that by taking the sum of all columns that correspond to a fixed effect of matrix X ends up to be a vector of all ones. This should hold for all fixed effects, because within a fixed effect all animals should be marked with a given level. Also the animals cannot be marked for multiple levels within a fixed effect. Thus the columns are orthogonal within a fixed effect. The rank of this matrix is 2 and has the following null space after Gaussian elimination:

$$\mathcal{N}(X) = \begin{bmatrix} 1 & 1 & -1 & -1 \end{bmatrix}^T$$

Assuming $R = I_{n,n}$, then the first block-matrix entry is given by:

$$X^T X$$

Then the matrix $X^T X$ has the following form:

$$X^T X = \begin{bmatrix} 3 & 0 & 3 & 0 \\ 0 & 3 & 2 & 1 \\ 3 & 2 & 5 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Each entry of the matrix shows the amount of animals with a combination of 2 fixed effects with a given a level, except for the diagonal entries, that numbers represent the amount of animals for 1 fixed effect with a given level. Therefore, a value zero means there are no animals with that combination of fixed effects. For example the entry in the first row and second column represents the value that a cow is a male and a female, since these properties belong to the same fixed effect the value is 0.

Note that $X^T Xy = 0$ if $y \in \mathcal{N}(X)$ and $Xy = 0$ if $y \in \mathcal{N}(X^T X)$. Thus $X^T X$ has the same null space as X. This also holds for any positive-definite (co)variance matrix R, therefore $\mathcal{N}(X) = \mathcal{N}(X^T R^{-1} X)$. If R is semi-positive definite, then R is singular. Hence the null space of $X^T R^{-1} X$ is defined by $\mathcal{N}(R)$ and $\mathcal{N}(X)$, which leads to $\mathcal{N}(X) \subseteq \mathcal{N}(X^T R^{-1} X)$.

Before generalising the null space in the previous example to $n_F > 1$ fixed effects, the following function has to be defined:

$$f_l(x) = \begin{cases} \begin{pmatrix} -1 & -1 & \dots & -1 \end{pmatrix} & x = 1 \\ \begin{pmatrix} \frac{1}{n_F - 1} & \frac{1}{n_F - 1} & \dots & \frac{1}{n_F - 1} \end{pmatrix} & else \end{cases}$$

where the vector size equals l. Let X be an arbitrary chosen incidence matrix, the vectors in the null space of X have the following form:

$$\begin{bmatrix} f_{l_1}(x_1) & f_{l_2}(x_2) & \cdots & f_{l_{n_F}}(x_{n_F}) \end{bmatrix}^T$$

where $x_1, x_2, \dots, x_{n_F}$ are chosen such that $x_i = 1$ and $x_j = 0 \ \forall i \neq j$ with $i, j \in \{1, 2, \dots, n_F\}$. Moreover, the number of levels for a fixed effect are given by $l_1, l_2, \dots, l_{n_F}$. Thus by construction there are $n_F - 1$ independent vectors with this particular form. Therefore the dimension of the null space of X is at least $n_F - 1$. This is due to the fact that a column of X corresponding to a fixed effect can be a linear combination of the other columns. It is worth to mention that $X^T X$ is a non-singular matrix if and only if $n_F = 1$.

6.4.3 Incidence matrix of animal effects W

The incidence matrix of animal effects W is of dimension $\mathbb{R}^{n_T \times n}$. Thus the rows consists of all animals with a (phenotypic) record and the columns consists of all animals. Note that $n \geq n_T$. To construct this matrix some additional data is needed from the cow population. For the sake of simplicity, table 6.2 is has been adjusted with the parents of the corresponding animal and the milk yield.

Table 6.3: The milk yied is used for animal effects.

Cow	Male	Female	Dam	Sire	Milk yield / year
1	1	0	7	9	N/A
2	1	0	7	9	N/A
3	0	1	7	9	33.3
4	0	1	8	9	30.5
5	1	0	8	9	N/A
6	0	1	8	N/A	31.7

The corresponding incidence matrix relating to all animals with a record of milk yield can be defined as:

$$W = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

The rows belong to the animals with a record and the columns belong to all animals. Thus the columns containing only zeroes belong to the animals without any record of milk yield. In this case the males and parents have no record of milk yield.

The matrix W is required for the computation of the coefficient matrix in the ssSNPBLUP model (6.26). The block-matrices containing W are $X_n^T R_n^{-1} W_n$, $X_g^T R_g^{-1} W_g$, $W_n^T R_n^{-1} X_n$, $W_g^T R_n^{-1} X_g$, $W_n^T R_n^{-1} W_n$ and $W_g^T R_n^{-1} W_g$. These matrices are easy to obtain, because the matrix X, W and R^{-1} are sparse and do not have a complex structure, as can be seen from this example and the previous paragraphs.

Note that W does not necessarily have full column or row rank, as can be seen from the example. This means that the block-matrices on the diagonal of the coefficient matrix in the ssSNPBLUP model (6.26) are singular. However, these block-matrices require an addition with $\Sigma_{11} = A^{nn}$ and $\Sigma_{22} = A^{gg} + \left(\frac{1}{w} - 1\right) A_{gg}^{-1}$. By definition A_{gg}, A^{nn} and A^{gg} are non-singular. Thus the mentioned block matrices on the diagonal are non-singular.

6.4.4 (Co)variance matrix Σ

In the previous paragraph the following matrix has been mentioned:

$$\begin{aligned}\Sigma^{-1} &= \begin{bmatrix}\Sigma^{11} & \Sigma^{12} & \Sigma^{13}\\ \Sigma^{21} & \Sigma^{22} & \Sigma^{23}\\ \Sigma^{31} & \Sigma^{32} & \Sigma^{33}\end{bmatrix}\\ &= \sigma_u^{-2}\begin{bmatrix}A^{nn} & A^{ng} & 0\\ A^{gn} & A^{gg}+\left(\frac{1}{w}-1\right)A_{gg}^{-1} & -\frac{1}{w}A_{gg}^{-1}Z\\ 0 & -\frac{1}{w}Z^TA_{gg}^{-1} & B^{-1}+\frac{1}{w}Z^TA_{gg}^{-1}Z\end{bmatrix}\end{aligned}$$

This matrix contains the inverse of the pedigree relationship matrix $A^{-1} \in \mathbb{R}^{n\times n}$, the inverse of the genotyped part of the pedigree relationship matrix $A_{gg}^{-1} \in \mathbb{R}^{n_g\times n_g}$ and the SNP genotype matrix $Z \in \mathbb{R}^{n_g\times m}$.

The inverse of the pedigree relationship matrix is defined by:

$$A^{-1} = \begin{bmatrix}A^{nn} & A^{ng}\\ A^{gn} & A^{gg}\end{bmatrix}$$

The matrix can be constructed by applying Henderson's rule [9]. The matrix is sparse and symmetric, see for details [9] and [24]. Thus the block-matrices A^{nn}, A^{ng}, A^{gn} and A^{gg} are sparse. In practice, this matrix can be very large ($> 10^6$ rows/columns). This yields at least 10^{12} entries for the matrix. In the past, difficulty arises in insufficient Random Access Memory (RAM) storage, because of the large amount of non-zero elements. Nowadays, computers have sufficient RAM to store A^{-1} [24]. Thus different approaches can be taken, which can take advantage of the BLAS routines that can handle sparse matrices efficiently.

The inverse of the pedigree relationship matrix concerning the genotyped animals, A_{gg}^{-1} is a dense matrix. The following relation can be used to calculate A_{gg}^{-1} [10][24][17]:

$$A_{gg}^{-1} = A^{gg} - A^{gn}(A^{nn})^{-1}A^{ng}$$

Remember that the matrices on the right-hand side are sparse matrices. Thus A_{gg}^{-1} is not expensive to compute.

In the ssSNPBLUP model proposed by Liu et al., the matrices involving the centred SNP genotype matrix Z takes the most demand for computing power. This demand is caused by the size of this matrix and it is usually not sparse. This matrix can be stored using 2 bits for each entries [24], since there are 4 different values: 0 for homozygous genotype, 1 for heterozygous genotype, 2 for alternate homozygous genotype and missing. This method is also known as Plink 1 binary form [4]. Note that the matrix entries corresponding to missing are set to 0 after centring.

6.5 Subdomain deflation method applied on SNP

In mathematics a subdomain is usually related to a space with a specific dimension. For example in $\mathbb{R}$ one can think about lines or in $\mathbb{R}^2$ one can think about the area. A set of

SNPs can be considered as a one dimensional space. Therefore, applying the subdomain deflation on SNPs can be done with the same approach for a 1 dimensional space (see chapter 5.4 for an example).

6.5.1 Subdomain decomposition with random sampling

The DPCG application on the ssSNPBLUP model [25] constructs subdomains with a specific size that contains SNPs. The SNPs are taken at random with a uniform distribution from the set of all SNPs. When a SNP has been chosen, the same SNP cannot be chosen again, this is also known as sampling without replacement.

For example take a set of 10 SNPs. Set the subdomain size to $l = 4$. Then there are 3 subdomains, with two of size 4 and one of size 2. The SNPs are chosen at random. For the sake of this example let the subdomains be defined as:

$$\Omega_1 = \{1, 4, 7, 8\}$$
$$\Omega_2 = \{3, 5, 9, 10\}$$
$$\Omega_3 = \{2, 6\}$$

where the number denote a SNP with that given index. The deflation-subspace matrix can be defined as:

$$Z = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

Note that the amount of subdomains depends on the size l, which results into the following formula:

$$\text{Number of subdomains} = \left\lceil \frac{\text{Total SNPs}}{l} \right\rceil$$

CHAPTER 7

Subdomain decomposition strategies

7.1 Data

To make use of the ssSNPBLUP method, information of the selected animals are needed. These are the fixed effects, phenotype, genotype and pedigree information. A small data set will be used at first. If the results are promising, then the data set will scale up. The data is simulated by using the program QMSim, where the set of parameters are explained in [3]. For the data set the following parameters have been used:

- 1 trait
- 5 chromosomes
- initial male population = 50
- initial female population = 14950
- Residual variance $var(e) = 0.7$
- Variance of the additive genetic effects $var(g) = 0.3$

This resulted in 164500 animals, where 18768 animals are genotyped. Moreover, there are 13100 SNPs. For creating the coefficient matrix of the ssSNPBLUP model, the SNPs have been filtered based on the minor allele frequency. Only SNPs with frequency $\alpha < p < 1 - \alpha$ with $\alpha = 0.01$ have been used. This filtering yields 12346 SNPs. The coefficient matrix has dimensions 176847×176847. Moreover, approximately 2.911% elements are non-zero elements. Henceforth, this matrix can be considered sparse.

7.2 Julia

The calculations are performed with the programming language *Julia*. This is a dynamic programming language on a high-level which is focused on high-performance. Julia supports elementary matrix and vector operations using the library *LinearAlgebra*. It also allows to work with sparse matrices using the library *SparseArrays*. The default sparse format is Compressed Sparse Column (CSC). This format allows for efficient arithmetic operations, column slicing and matrix-vector products. The CSC format consists of a vector V containing all the non-zero values, a vector R containing the row indices corresponding to the non-zero values and a vector C contains the amount of entries in a column, which is summed with the previous entries. An example will be given below on how to convert a matrix to CSC format.

$$A = \begin{bmatrix} 3 & 0 & 2 \\ 0 & 4 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The CSC format is given by:

$$\begin{aligned} V &= \begin{pmatrix} 3 & 4 & 1 & 2 \end{pmatrix} \\ R &= \begin{pmatrix} 0 & 1 & 2 & 0 \end{pmatrix} \\ C &= \begin{pmatrix} 0 & 1 & 3 & 4 \end{pmatrix} \end{aligned}$$

Note that the first row starts with index 0 instead of 1 in this example. Reading the vectors will lead to:

$$\begin{aligned} \text{First column: } & V[C[0]:C[1]] = 3 \text{ and } R[C[0]:C[1]] = 0 \\ \text{Second column: } & V[C[1]:C[2]] = (4,1) \text{ and } R[C[1]:C[2]] = (1,2) \\ \text{Third column: } & V[C[2]:C[3]] = 2 \text{ and } R[C[2]:C[3]] = 0 \end{aligned}$$

7.3 Deflation operator applied on the ssSNPBLUP related with ssGBLUP

The linear system used in the ssSNPBLUP model is shown in the following equation:

$$Cx = b$$

with

$$C = \begin{bmatrix} X^T R^{-1} X & X_n^T R_n^{-1} W_n & X_g^T R_g^{-1} W_g & 0 \\ W_n^T R_n^{-1} X_n & W_n^T R_n^{-1} W_n + \Sigma^{11} & \Sigma^{12} & \Sigma^{13} \\ W_g^T R_g^{-1} X_g & \Sigma^{21} & W_g^T R_g^{-1} W_g + \Sigma^{22} & \Sigma^{23} \\ 0 & \Sigma^{31} & \Sigma^{32} & \Sigma^{33} \end{bmatrix}$$

$$x = \begin{bmatrix} \hat{\beta} \\ \hat{u}_n \\ \hat{u}_g \\ \hat{g} \end{bmatrix}, \; b = \begin{bmatrix} X^T R^{-1} y \\ W_n^T R_n^{-1} y_n \\ W_g^T R_g^{-1} y_g \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} \Sigma^{11} & \Sigma^{12} & \Sigma^{13} \\ \Sigma^{21} & \Sigma^{22} & \Sigma^{23} \\ \Sigma^{31} & \Sigma^{32} & \Sigma^{33} \end{bmatrix} = \sigma_u^{-2} \begin{bmatrix} A^{nn} & A^{ng} & 0 \\ A^{gn} & A^{gg} + \left(\frac{1}{w} - 1\right) A_{gg}^{-1} & -\frac{1}{w} A_{gg}^{-1} Z \\ 0 & -\frac{1}{w} Z^T A_{gg}^{-1} & \frac{k}{1-w} I_{m,m} + \frac{1}{w} Z^T A_{gg}^{-1} Z \end{bmatrix}$$

Without loss of generality, $\sigma_u^2 = 1$ will be used from now on. Moreover, the Woodbury matrix identity ([7], chapter 2.1.4) will be used at some point, which holds the following equality:

$$(A + UCV)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

where A, C, U and V are conformable matrices. The latter definition means that the dimensions of the matrices are suitable with one and another in this equation. By applying the deflation subspace method with 1 SNP per subdomain on the linear system (see chapter 6.5) , a relation between the ssSNPBLUP and ssGBLUP model can be seen ([25], see Appendix 1). The derivation of this relation is analogous to [25] Appendix 1. Consider the linear system of equations:

$$Cx = b \iff \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \text{ with } C_{11} = \begin{bmatrix} X^T R^{-1} X & X_n^T R_n^{-1} W_n & X_g^T R_g^{-1} W_g \\ W_n^T R_n {-1} X_n & W_n^T R_n^{-1} W_n + \Sigma^{11} & \Sigma^{12} \\ W_g^T R_g^{-1} X_g & \Sigma^{21} & W_g^T R_g^{-1} W_g + \Sigma^{22} \end{bmatrix},$$

$$C_{12} = \begin{bmatrix} 0 \\ \Sigma^{13} \\ \Sigma^{23} \end{bmatrix}, C_{21} = \begin{bmatrix} 0 & \Sigma^{31} & \Sigma^{32} \end{bmatrix} \text{ and } C_{22} = \Sigma^{33}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \text{ with } x_1 = \begin{bmatrix} \hat{\beta} \\ \hat{u}_n \\ \hat{u}_g \end{bmatrix}, x_2 = \hat{g}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \text{ with } b_1 = \begin{bmatrix} X^T R^{-1} y \\ W_n^T R_n^{-1} y_n \\ W_g^T R_g^{-1} y_g \end{bmatrix}, b_2 = 0$$

The deflation-subspace matrix F is defined by taking all non-SNP effects into 1 subdomain and each SNP effect has its own subdomain. Thus F has the following form:

$$F = \begin{bmatrix} \mathbf{e} & 0 \\ 0 & I_{22} \end{bmatrix}$$

where $\mathbf{e} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}^T$ and I_{22} is the identity matrix. Note that the identity matrix can be taken, since every column that represents a subdomain has one non-zero entry. Let the deflation matrix P be defined as:

$$P = I - CFE^{-1}F^T$$

where the Galerkin matrix is $E = F^T C F$ and the inverse is given by:

$$E^{-1} = \begin{bmatrix} Q & -Q\mathbf{e}^T C_{12} C_{22}^{-1} \\ -C_{22}^{-1} C_{21}\mathbf{e}Q & T \end{bmatrix}$$

with

$$Q = \left(\mathbf{e}^T \left(C_{11} - C_{12}C_{22}^{-1}C_{21}\right)\mathbf{e}\right)^{-1}$$
$$T = C_{22}^{-1} + C_{22}^{-1}C_{21}\mathbf{e}Q\mathbf{e}^T C_{12}C_{22}^{-1}$$

see Appendix 10.2 for the derivation of the inverse.

Using the result from appendix 10.2 with $Z_1 = \mathbf{e}$ and $Z_2 = \mathbf{I}_{22}$. The deflated coefficient matrix C can now be formulated as:

$$PC = \begin{bmatrix} S - S\mathbf{e}(\mathbf{e}^T S\mathbf{e})^{-1}(S\mathbf{e})^T & 0 \\ 0 & 0 \end{bmatrix} \tag{7.1}$$

with

$$S = C_{11} - C_{12}C_{22}^{-1}C_{21}$$

The latter matrix can be reformulated as:

$$S = \begin{bmatrix} X^T R^{-1} X & X_n^T R_n^{-1} W_n & X_g^T R_g^{-1} W_g \\ W_n^T R_n^{-1} X_n & W_n^T R_n^{-1} W_n + A^{nn} & A^{ng} \\ W_g^T R_g^{-1} X_g & A^{gn} & W_g^T R_g^{-1} W_g + A^{gg} + (\frac{1}{w} - 1)A_{gg}^{-1} - \Sigma^{23}(\Sigma^{33})^{-1}\Sigma^{32} \end{bmatrix}$$

where

$$\Sigma^{23}(\Sigma^{33})^{-1}\Sigma^{32} = \frac{1}{w}A_{gg}^{-1} - G_{gg}^{-1}$$

In the last equation, the Woodbury matrix identity has been applied, where

$$G_{gg} = Z\frac{1-w}{k}I_{m,m}Z^T + wA_{gg}$$

Substituting this result yields:

$$S = \begin{bmatrix} X^T R^{-1} X & X_n^T R_n^{-1} W_n & X_g^T R_g^{-1} W_g \\ W_n^T R_n^{-1} X_n & W_n^T R_n^{-1} W_n + A^{nn} & A^{ng} \\ W_g^T R_g^{-1} X_g & A^{gn} & W_g^T R_g^{-1} W_g + A^{gg} + G_{gg}^{-1} - A_{gg}^{-1} \end{bmatrix} \tag{7.2}$$

The coefficient matrix seen in equation (7.2) is the same coefficient matrix shown in (6.23). The latter coefficient matrix (6.23) is the one used in the ssGBLUP model. Therefore the following equality holds:

$$C^* = S$$

with

$$C^* = \begin{bmatrix} X^T R^{-1} X & X_n^T R_n^{-1} W_n & X_g^T R_g^{-1} W_g \\ W_n^T R_n^{-1} X_n & W_n^T R_n^{-1} W_n + A^{nn} & A^{ng} \\ W_g^T R_g^{-1} X_g & A^{gn} & W_g^T R_g^{-1} W_g + A^{gg} + G_{gg}^{-1} - A_{gg}^{-1} \end{bmatrix}$$

Note that the coefficient matrix C^* does take into account SNP effects. The SNP effects are included in the coefficient matrix G_{gg}^{-1}, which is a dense matrix. This can also cause memory issues when there are more than 100000 genotyped animals [12]. The ssSNPBLUP avoids using the inverse of G_{gg} explicitly.
The first block-matrix entry in the deflated coefficient matrix PC (7.1) only differs from C^* due to the term $S\mathbf{e}(\mathbf{e}^TS\mathbf{e})^{-1}(S\mathbf{e})^T$. This term can be rewritten as:

$$S\mathbf{e}(\mathbf{e}^TS\mathbf{e})^{-1}(S\mathbf{e})^T = \left(\sum_{i=1}^{n_T}\sum_{j=1}^{n_T} s_{ij}\right)^{-1} S\mathbf{e}(S\mathbf{e})^T$$

The matrix-vector multiplication $S\mathbf{e}$ yields a vector that contains all the row sums of S. These multiplications are then scaled by the total summation of all entries of S.
The following holds for the deflated coefficient matrix C:

$$PC = \begin{bmatrix} C^* & 0 \\ 0 & 0 \end{bmatrix} - \begin{bmatrix} \left(\sum_{i=1}^{n_T}\sum_{j=1}^{n_T} s_{ij}\right)^{-1} S\mathbf{e}(S\mathbf{e})^T & 0 \\ 0 & 0 \end{bmatrix} \tag{7.3}$$

Hence the deflation operator applied on the linear system of the ssSNPBLUP model results into a linear system that contains the coefficient matrix C^*. The latter is the coefficient matrix used for the ssGBLUP model. If one were to use the CG method applied on the deflated linear system PC, then the inverse of the genomic relationship G_{gg}^{-1} has to be calculated, which is included in the coefficent matrix C^*. The inverse of the genomic relationship matrix is a dense matrix and causes memory issues when the group of genotyped animals is large. Thus this is not a favourable option. The DCG method applied on C does not contain the inverse of the genomic relationship matrix. However, to perform the iteration process it uses a large Galerkin matrix, since the subdomain size is 1. This makes each iteration more expensive. Thus the DCG method with a small subdomain size is also not favourable. Therefore, finding larger subdomains that approximates the eigenvectors corresponding to the problematic eigenvalues, can be of great value for the convergence of the ssSNPBLUP model. It is worth mentioning that this result can also be extended for a preconditioned linear system with ease.

7.4 Subdomain selection strategy

7.4.1 Subdomain selection per chromosome

The effectiveness of the deflation method depends on how well the span of the columns of the deflation-subspace matrix Z approximate the eigenvectors that correspond to the problematic eigenvalues. In [25] it is stated that the large eigenvalues are in some way related with the SNP effects. Hence the computational domain is divided into subdomains based on the SNPs per trait.

Neigbouring SNPs can predict the outcome of other SNPs around them and of themselves [27]. Moreover, SNP effects are correlated with each other. *Hence, a grouping of SNP effects can be created that are relatively highly correlated and share similar correlations with other SNPs.*

For the following figures, the mean sum squared correlation of a subdomain χ is required, which is defined as:

$$\frac{1}{|\chi|}\sum_{x\in\chi}\rho_x^2$$

where $|\chi|$ denotes the amount of elements contained in χ and ρ_x is the correlation given by the correlation matrix. To get an impression of the correlation of the SNP effects per chromosome, the following figures have been created:

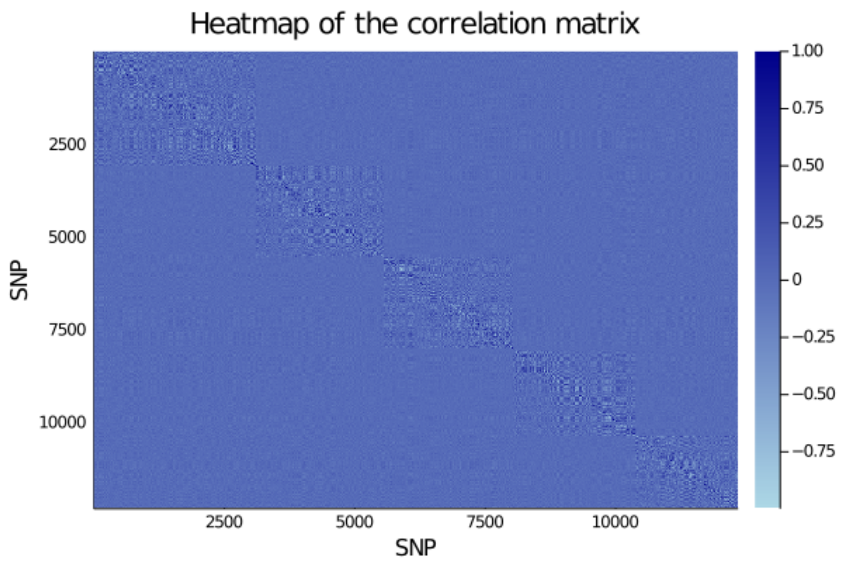


Figure 7.1: The correlation of each SNP effect.

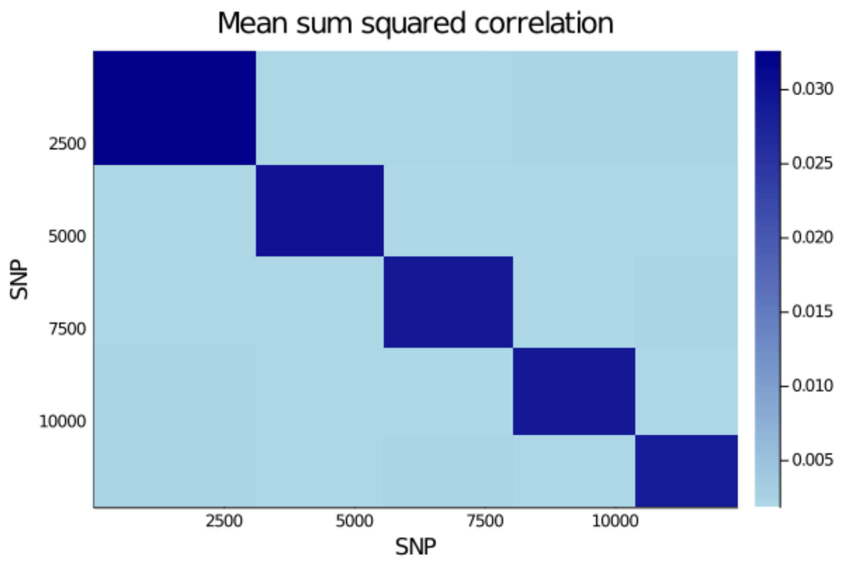


Figure 7.2: The mean sum squared correlation of each SNP effect grouped per chromosome.

From figure 7.1 it can be seen that the correlations are larger for SNP effects corresponding to the same chromosome. This pattern can easily be seen in figure 7.2.

This leads to the motivation to construct the deflation-subspace matrix per chromosome. There are 5 columns dedicated for each chromosome. The first column is the subdomain containing all non-SNP effects. Thus in this case there are 6 columns for the deflation-subspace matrix.

The performance of this method will be compared with the PCG method using diagonal scaling and the ssGBLUP model using the PCG method with diagonal scaling. It is worth to mention that the linear system from the ssGBLUP model can be obtained through the deflation method. This can be done by selecting the subdomain size by 1 [25]. Hence every SNP effect has its own column in the deflation-subspace matrix. Since this is the smallest achievable size, the convergence of the ssGBLUP model can be seen as a lower bound.

The following stopping criterion will be used:

$$\frac{||r||_2}{||b||_2} < \epsilon$$

where $\epsilon = 10^{-6}$, r is the residual obtained from the CG method and b is the right hand-side of the respective model. A maximum of 2000 iterations will be performed excluding the initial iteration. It is worth to mention that the results are a little bit sensitive to rounding errors. This is due to parallelisation and the implementation of certain functions like the dot product. As a consequence the iterations can vary by 1-10 iterations depending on the implementation of those certain functions.

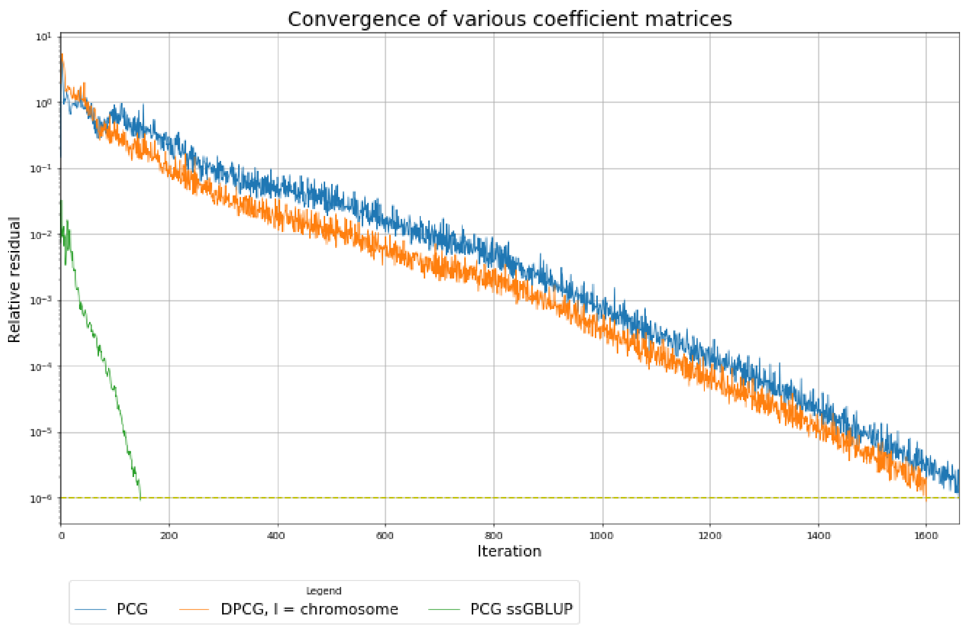


Figure 7.3: The convergence of the DPCG method where each subdomain is divided per chromosome.

In figure 7.3 it can be seen that the PCG and DPCG method for the ssSNPBLUP model has converged after 1662 and 1602 iterations, respectively. Whilst the ssGBLUP model has converged within 149 iterations. The PCG and DPCG method for the ssSNPBLUP model both follow a similar pattern. After iteration 76, the DPCG method starts with a smaller value than the PCG used for the ssSNPBLUP model and keeps this trend. Thus the DPCG method converges faster than the PCG method.

The Ritz values of the different coefficient matrices is shown in table 7.1.

Table 7.1: The smallest and largest Ritz values of the preconditioned matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG	0.001435	323.05	226199.1	1662
DPCG	0.001500	322.51	214956.3	1602
ssGBLUP	0.001799	2.47	1370.7	149

7.4.2 Subdomain selection on groups of SNPs within a chromosome

Since constructing the subdomains per chromosome improved the convergence of the DPCG method. The convergence can be improved further by taking smaller subdomains.

For each chromosome the subdomains are constructed by sampling SNP effects without replacement. For the sampling, the same seed has been used for all subdomain sizes. Therefore, the same SNP cannot be chosen again. Define l as the size of the subdomain and c the number of chromosomes. Thus each subdomain holds l SNP effects. The total amount of subdomains depend on the amount of SNPs per chromosome. Define s_{c_i} as the number of SNPs for the i-th chromosome. The amount of subdomains within the i-th chromosome equals $\lceil \frac{s_{c_i}}{l} \rceil$. Thus the total amount of subdomains equals $1 + \sum_{i=i}^{c} \lceil \frac{s_{c_i}}{l} \rceil$, where the additional subdomain is added from the non-SNP effects.

Different subdomain sizes have been chosen, which are $l = \{1, 3, 10, 50, 100\}$. The convergence of this selection strategy can be seen in figure 7.4.

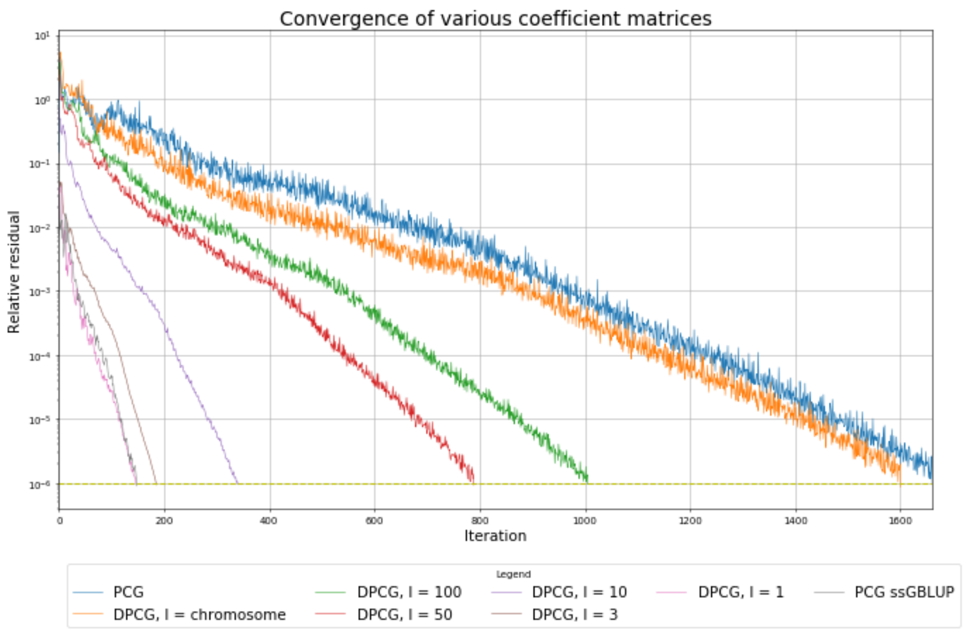


Figure 7.4: The convergence of the DPCG method where each subdomain contains sampled SNPs corresponding to a chromosome.

From figure 7.4 it is clear that a smaller size for each subdomain results in an improvement of the convergence. The convergence patterns of all subdomain sizes have 2 distinct slope patterns in common, where the slope becomes steeper halfway during the iteration process.
For $l = 1$ and $l = 3$, the graphs indicate superlinear convergence. Also the convergence patterns are very similar to the ssGBLUP model. This convergence behaviour can be expected for $l = 1$, because the used deflated preconditioned system (7.3) is similar to the preconditioned system of ssGBLUP. Moreover, this could mean that the SNP effects in the ssSNPBLUP model causes larger eigenvalues than the ssGBLUP model, which has also been described in [25]. It seems that for a small subdomain size around 3 yields for the best improvement. Although, the amount of additional work per iteration has to be taken into consideration compared with a larger subdomain size, because the deflation-subspace matrix is larger for smaller subdomain sizes. Noticeably, in [25] the convergence of the DPCG method yielded drastic improvements for small subdomain sizes ≤ 10. Although, the DPCG method is applied on a slightly different ssSNPBLUP model proposed by Mäntysaari and Strandén.

Table 7.2: The smallest and largest Ritz values of the deflated preconditioned matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG	0.001435	323.05	226199.1	1662
DPCG	0.001500	322.51	214956.3	1602
DPCG $l = 100$	0.001498	82.92	55371.7	1007
DPCG $l = 50$	0.001498	51.32	34260.8	790
DPCG $l = 10$	0.001500	7.30	4865.7	341
DPCG $l = 3$	0.001506	2.30	1525.9	186
DPCG $l = 1$	0.001524	2.30	1507.5	147
ssGBLUP	0.001799	2.47	1370.7	149

If $l = 1$, then the columns of the deflation-subspace matrix corresponding to the SNP effects should span any possible combinations encountered for $l > 1$. Note that the selection of SNPs has been done randomly. Thus $l = 1$ is the largest convergence gain that can be obtained with this deflation method, which can also be seen from figure 7.4.

7.4.3 Different initial solution with random sampling

From figure 7.4, it is quite noticeable that for subdomain sizes $l = \{1, 3\}$ compared with the other subdomain sizes, that the relative residual starts with a small value. This could be that the initial solution used, which is the vector with all its entries equal to 0, is close to the solution of this linear system. This convergence behaviour can be explained, by changing the initial solution or comparing the initial solution with the approximated solution,

The same method used in chapter 7.4.2 to construct the subdomain will be applied in this chapter as well. The initial solution created will have all its entries sampled from the standard normal distribution $\mathcal{N}(0, 1)$. The same seed will be taken for each result. Hence all the initial solutions taken for different subdomain sizes will be the same. The initial solution different from the zero solution will be taken for the subdomain size $l = \{1, 3, 10, 50, 100\}$.

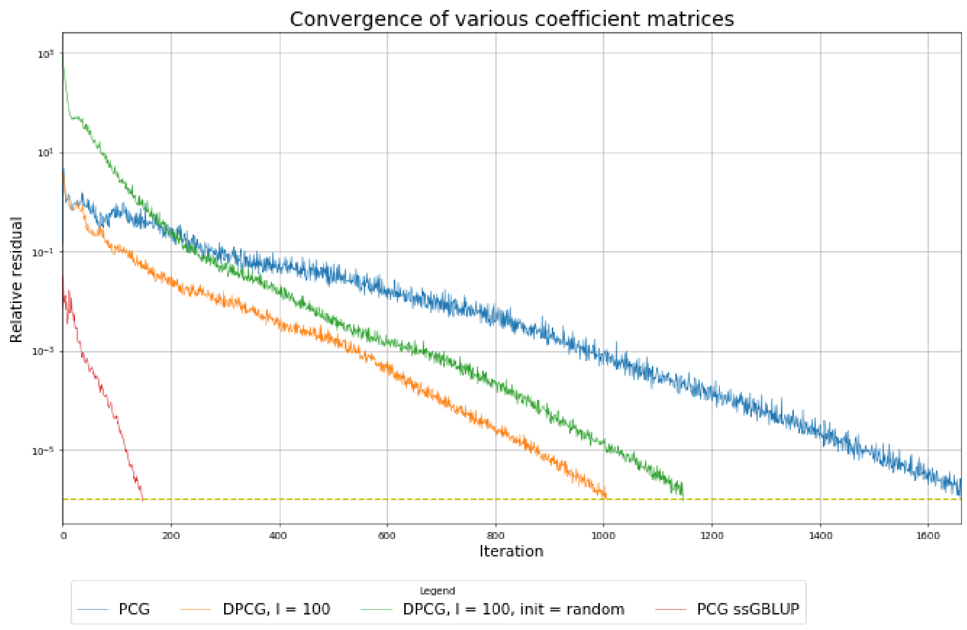


Figure 7.5: The convergence of the DPCG method where each subdomain contains sampled SNPs corresponding to a chromosome. The subdomain size $l = 100$. In the figure two different initial solutions are compared.

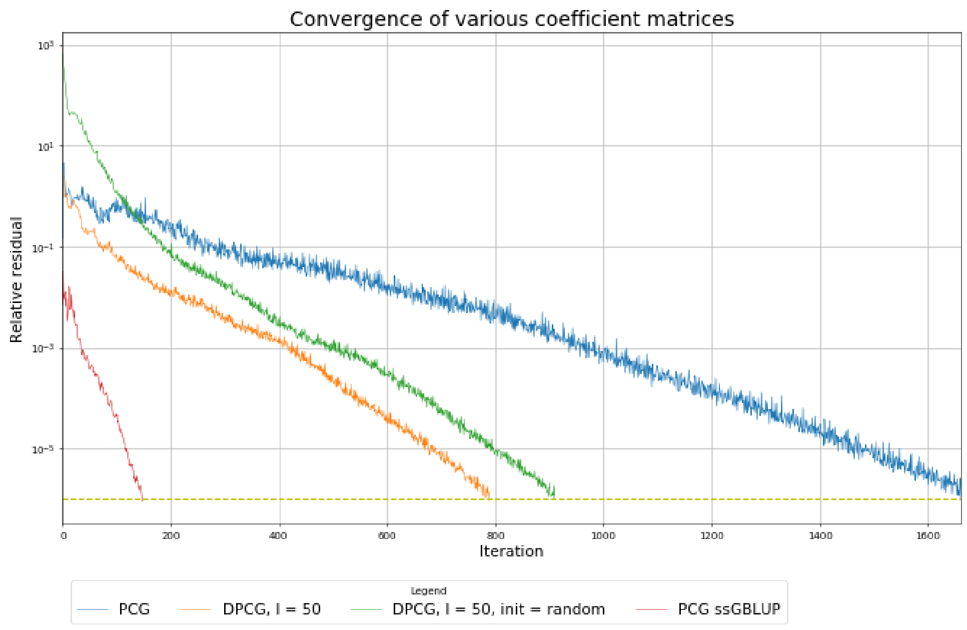


Figure 7.6: The convergence of the DPCG method where each subdomain contains sampled SNPs corresponding to a chromosome. The subdomain size $l = 50$. In the figure two different initial solutions are compared.

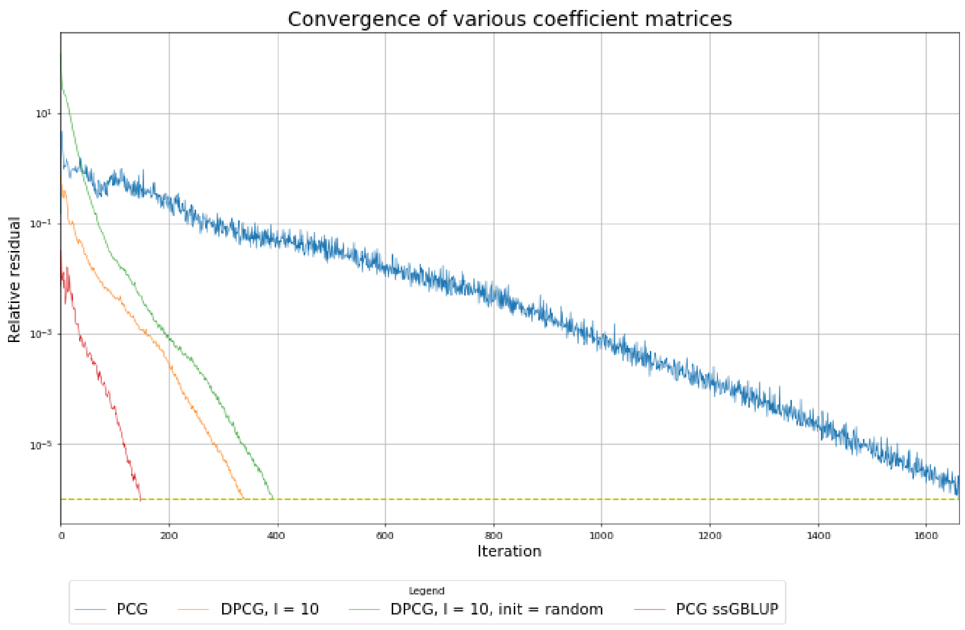


Figure 7.7: The convergence of the DPCG method where each subdomain contains sampled SNPs corresponding to a chromosome. The subdomain size $l = 10$. In the figure two different initial solutions are compared.

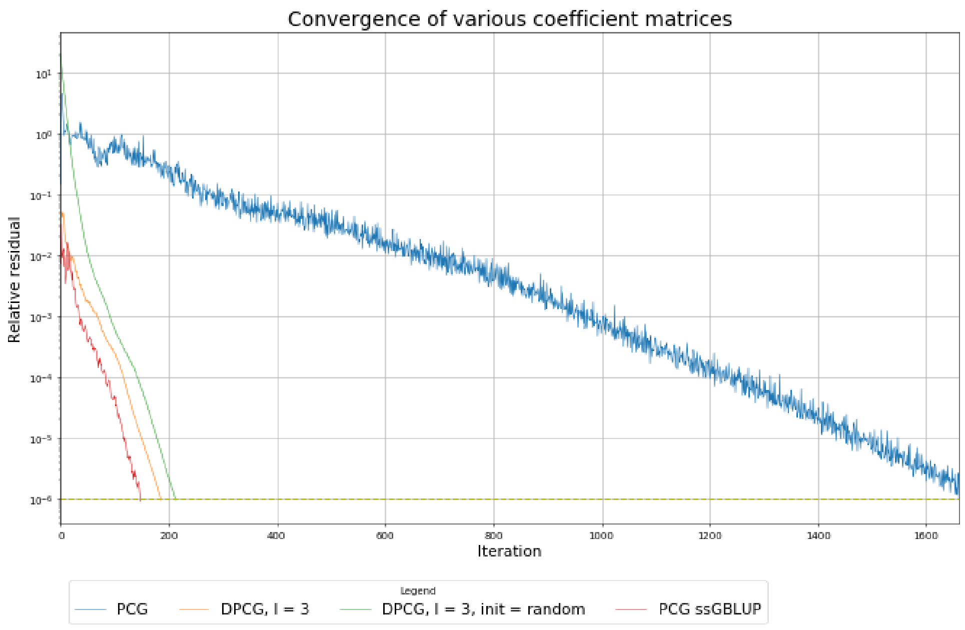


Figure 7.8: The convergence of the DPCG method where each subdomain contains sampled SNPs corresponding to a chromosome. The subdomain size $l = 3$. In the figure two different initial solutions are compared.

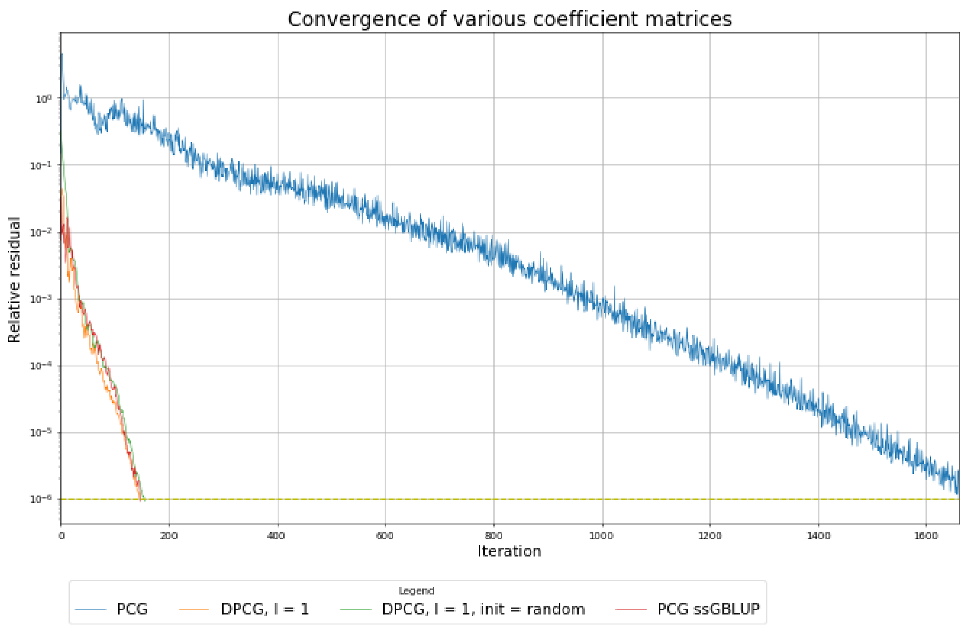


Figure 7.9: The convergence of the DPCG method where each subdomain contains sampled SNPs corresponding to a chromosome. The subdomain size $l = 1$. In the figure two different initial solutions are compared.

Table 7.3: The smallest and largest Ritz values of the deflated preconditioned matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
DPCG $l = 100$	0.001498	82.92	55371.7	1007
DPCG initial $l = 100$	0.001498	82.92	55369.1	1149
DPCG $l = 50$	0.001498	51.32	34260.8	790
DPCG initial $l = 50$	0.001498	51.32	34259.2	911
DPCG $l = 10$	0.001500	7.30	4865.7	341
DPCG initial $l = 10$	0.001500	7.30	4865.1	393
DPCG $l = 3$	0.001506	2.30	1525.9	186
DPCG initial $l = 3$	0.001506	2.30	1525.6	214
DPCG $l = 1$	0.001524	2.30	1507.5	147
DPCG initial $l = 1$	0.001523	2.30	1508.8	156

From figures 7.5 to 7.9, it can be observed that the relative residual starts of with a larger value using a different initial solution compared with the zero solution. By comparing each subdomain separately, it takes more iterations to converge if the initial vector is not the zero vector as the initial solution. Nonetheless, the convergence behaviour behaves the same as using the zero vector as an initial solution. It is worth to mention that the entries of the solution are close to zero, because the SNP effects take on values close to zero.

7.4.4 Map of the chromosome

In the previous section a random sampling of SNPs within a chromosome has been used. This resulted in an improvement of convergence, whenever the subdomain size decreases. A smaller subdomain size is not favourable, because the DPCG method becomes more expensive in return.
From the previous results, dividing the SNPs per chromosome yielded in an improvement of convergence. The decision is based on the relative high correlations between SNPs within a chromosome and low correlations with SNPs from two different chromosomes. Noticeably, random sampling disregards the notion of using correlations of SNPs. Thus the question arises, *if constructing subdomains with relative high correlated SNPs improves the convergence.*
The map of the chromosome is simply taking the first SNP, then the second SNP which is a neighbour of the first SNP, then the third SNP which is a neighbour of the second SNP and so on, see figure 7.10 for a simple illustration.

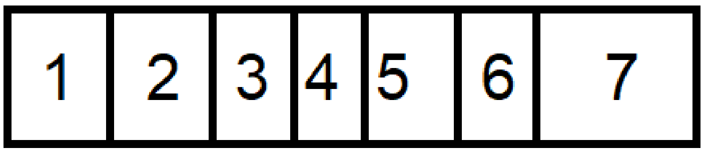


Figure 7.10: A chromosome is a long DNA molecule and these consists of SNPs. A simplification yields an object with segments. Each segment corresponds to a SNP.

Usually, SNPs that are close to one another have a higher correlation than SNPs that are further away. Thus the map of the chromosome can be taken, since it is quite likely that it groups the SNPs with relative high correlation. This is a naive approach, because within the subdomains with fixed size multiple SNPs can be grouped that are not relatively highly correlated.
The simulated Bradford data has these SNPs already aligned in their natural order. Thus extracting the map can be done by following the indices in an ascending order. It is worth to mention that in practice the map information is not available.

The subdomains following the map are compared with the random sampling method. For the results the subdomain sizes $l = \{3, 10, 50, 100\}$ have been selected.

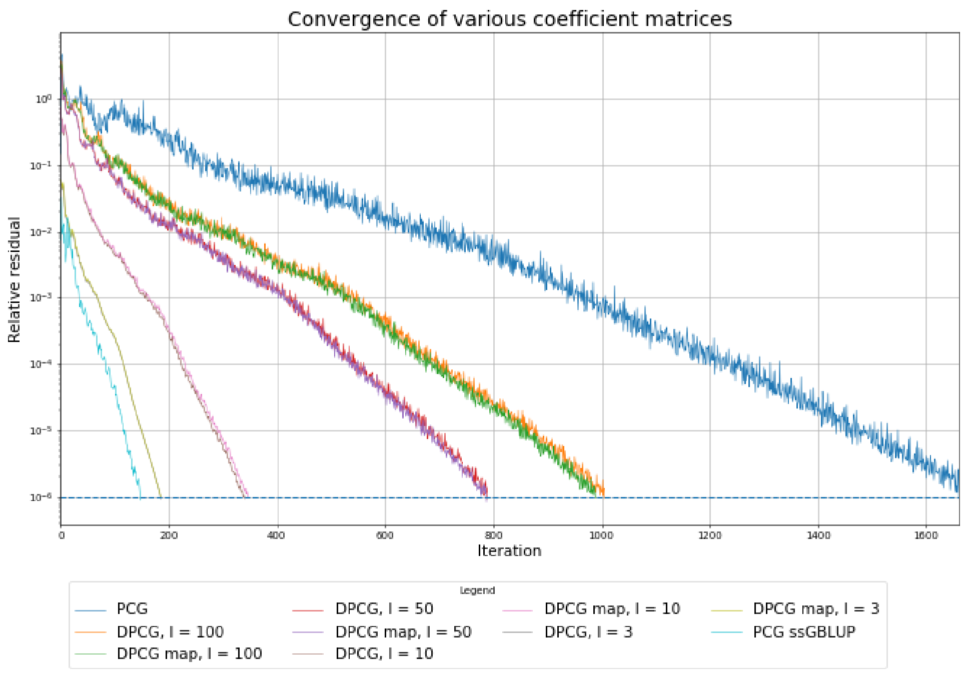


Figure 7.11: The convergence of the DPCG method where each subdomain is divided per chromosome and within the chromosomes the subdomains are divided into groups of SNPs with a fixed size following the map of the chromosome.

Table 7.4: The smallest and largest Ritz values of the deflated preconditioned matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG	0.001435	323.05	226199.1	1662
DPCG $l = 100$	0.001498	82.92	55371.7	1007
DPCG map $l = 100$	0.001498	73.20	48879.5	990
DPCG $l = 50$	0.001498	51.32	34260.8	790
DPCG map $l = 50$	0.001498	50.43	33669.9	788
DPCG $l = 10$	0.001500	7.30	4865.7	341
DPCG map $l = 10$	0.001500	7.78	5186.9	347
DPCG $l = 3$	0.001506	2.30	1525.9	186
DPCG map $l = 3$	0.001506	2.30	1525.9	186
ssGBLUP	0.001799	2.47	1370.7	149

The convergence behaviour of following the map method and random sampling is identical to one another for $l = 3$, as can be seen in figure 7.11 and table 7.4. This is probably due to the fact that the columns of the deflation-subspace matrix approximate the eigenvectors of the problematic eigenvalues quite well, because the subdomain size is small.

It seems that the convergence behaviour of following the map method is almost the same as the random sampling method. However, taking the map for $l = 10$ has a slower convergence and for $l = \{50, 100\}$ the convergence is faster. This behaviour can be seen at the tail of the graphs and table 7.4. Thus from these results, following the map does not always yield an improvement. This could be due to the fact that subdomains can still contain SNPs that are not related with each other.

7.4.5 Random grouping of SNPs per subdomain without chromosome restriction

The subdomains in the previous section contain random sampled SNPs from the same chromosome. The motivation of this choice, has to do with the assumption that SNPs are more likely to have a higher correlation than with SNPs from different chromosomes. This does not hold in general. Henceforth, for the following results, the SNPs are taken at random without the restriction that they must be in the same chromosome.

The results of taking SNPs at random will be compared with the method described in 7.4.1 (SNPs taken at random with the restriction that they must be in the same chromosome). This subdomain selection strategy will be applied for $l = \{1, 3, 10, 50, 100\}$.

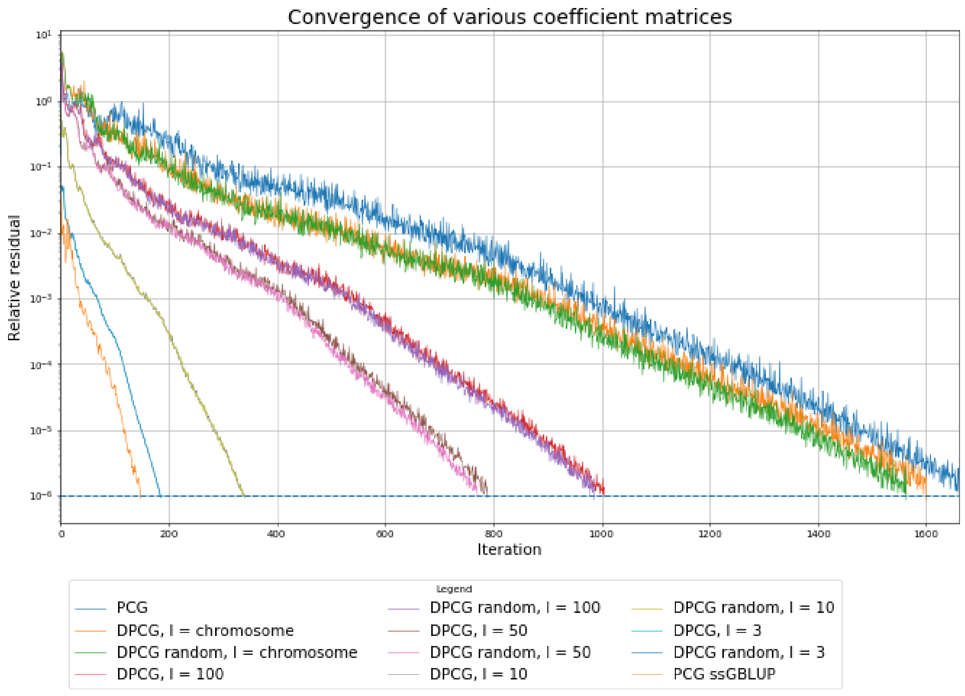


Figure 7.12: The convergence of the DPCG method where each subdomain with a fixed size contains a group of random SNPs without the chromosome restriction.

Table 7.5: The smallest and largest Ritz values of the deflated preconditioned matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG	0.001435	323.05	226199.1	1662
DPCG chromosome	0.001500	322.51	214956.3	1602
DPCG random chromosome	0.001497	266.50	177994.46	1565
DPCG $l = 100$	0.001498	82.92	55371.7	1007
DPCG random $l = 100$	0.001497	78.02	52102.9	988
DPCG $l = 50$	0.001498	51.32	34260.8	790
DPCG random $l = 50$	0.001498	41.46	27685.7	772
DPCG $l = 10$	0.001500	7.30	4865.7	341
DPCG random $l = 10$	0.001500	7.11	4738.3	341
DPCG $l = 3$	0.001506	2.30	1525.9	186
DPCG random $l = 3$	0.001505	2.30	1526.1	186
ssGBLUP	0.001799	2.47	1370.7	149

Thus for some subdomain sizes a small improvement in convergence can be seen. Since the SNPs per subdomain are taken at random, it could be that each subdomain contains SNPs that are more related with one and another. For small subdomain sizes $l < 10$ the convergence is very similar to the convergence with the chromosome restriction. Once again, this is probably due to the fact that the columns of the deflation subspace matrix for a small subdomain size is more likely to approximate the eigenvectors better than a large subdomain size. For the subdomain sizes $l = \{50, 100, chromosome\}$ the convergence is slightly faster without the restriction, which can also be observed from table 7.5. Whilst for a smaller subdomain size $l = \{3, 10\}$, the convergence rate is the same as selecting SNPs corresponding to a chromasome at random. Whenever the size is large, it is more likely that a subdomain contains SNPs that are not related with each other. This could also explain the convergence improvement for $l = \{50, 100, chromosome\}$.

7.4.6 k-means clustering

From the previous chapter 7.4.5, some improvements have been observed. This improvement relies on the hypothesis that the subdomains contain SNPs that affect the same gene/trait, thus they are related with each other. Although, it is quite difficult to group them based on the idea that they are related, because that information is unknown for most SNPs. However, the correlation of the SNPs is known and it is possible to group SNPs that share similar correlations with each other. For example, define the correlation matrix as:

$$COR = \begin{bmatrix} 1 & 0.9 & 0.3 & 0.3 \\ 0.9 & 1 & 0.5 & 0.5 \\ 0.3 & 0.5 & 1 & 0.5 \\ 0.3 & 0.5 & 0.5 & 1 \end{bmatrix}$$

This matrix shows the correlation between 4 different SNPs. The diagonal values shows the correlation of the SNPs with itself. A column shows the correlation of a SNP with other SNPs. The SNPs will be grouped based on the difference between each entry of a column, then this difference is summed and SNPs that have a small summed difference are grouped together. From inspection it can be seen that the first and second column share similar correlations. As well as the third and fourth colum. Thus 2 groups are made, which contain $\{SNP1, SNP2\}$ and $\{SNP3, SNP4\}$.

Something similar is done using the k-means clustering algorithm, also known as Lloyd's algorithm [11]. In general, the k-means clustering constructs k disjoint sets from n data points. Each set has a centre, this will be defined later on. For each set, the elements have in common that they are nearest to the centre of their group compared with other centres. This distance is based on the Euclidean distance, also referred as the 2-norm. Define $\chi \subset \mathbb{R}^d$ as the set of data points with $d > 0$. Thus these data points can be vectors with dimension d. Let k be the amount of clusters/groups. Let $\chi = \bigcup_{i=1}^{k} \chi_i$ where χ_i is a subset containing data points with $\chi_i \cap \chi_j = \emptyset$ if $i \neq j$. Define the centre as:

$$c_i = \frac{1}{|\chi_i|} \sum_{x \in \chi_i} x$$

with $|\cdot|$ denoting the size of the set. The k-means algorithm minimises the following cost function

$$\arg\min_{\chi} \sum_{i=1}^{k} \sum_{x \in \chi_i} \|x - c_i\|_2^2$$

The k-means algorithm is given by [2]:

Algorithm 5: k-means algorithm

```
1 Choose $k$ random initial centres $\mathcal{C} = \{c_1, c_2, \dots, c_k\}$ from $n$ data points.
2 for $l = 1, 2, \dots,$ until $\mathcal{C}$ no longer changing do
3     for $i = 1, 2, \dots, k$ do
4         Set the cluster $\chi_i$ to be the set of data points in $\chi$ that are closer to $c_i$
            than they are to $c_j$ for all $i \neq j$.
5         Set $c_i$ as the centre of $\chi_i$ by $c_i = \frac{1}{|\chi_i|} \sum_{x \in \chi_i} x$
```

The complexity of this algorithm is $\mathcal{O}(n^{kd})$ where $n = |\chi|$ is the number of data points. The initial centres are chosen at random from a uniform distribution. This can result in poor choices of initial centres, because it can happen that initial centres near each other are chosen. Moreover, the k-means algorithm does not guarantee convergence to the global minimum.

The k-means++ algorithm tries to remedy the poor choice of initial centres, where each initial centre has a given probability assigned. Define $D(x)$ as the shortest distance from a data point to the closest centre that is chosen. Then this probability/weight is defined as [2]:

$$\frac{D(x)^2}{\sum_{x\in\chi} D(x)^2}$$

Thus this weight takes into account the distance between other centres. Centres that are further away from each other, are more likely to be chosen at random.
The k-means++ algorithm is given by [2]:

Algorithm 6: k-means++ algorithm

```
1 Choose one initial centre $c_1$ uniformly at random from $\chi$.
2 for $i = 2, 3 \ldots, k$ do
3     Add a new centre $c_i$ from $\chi$ with probability $\frac{D(x)^2}{\sum_{x\in\chi} D(x)^2}$
4 for $l$ = 1,2,..., until $\mathcal{C}$ no longer changing do
5     for $i = 1, 2, \ldots, k$ do
6         Set the cluster $\chi_i$ to be the set of points in $\chi$ that are closer to $c_i$ than
            they are to $c_j$ for all $i \neq j$.
7         Set $c_i$ as the centre of $\chi_i$ by $c_i = \frac{1}{|\chi_i|}\sum_{x\in\chi_i} x$
```

The difference between computation of k-means and k-means++ is the initialisation of the initial centres. Note that for both algorithms the cluster sizes can be different for each cluster.

The k-means algorithm has been applied on the correlation matrix of SNPs to obtain groups of SNPs with similar correlation. These groups are used for the construction of the subdomains. The same seed is used for the following results, since the k-means algorithm takes the initial centres at random. Also the amount of subdomains should be similar with the previous methods. This can be done by applying the formula:

$$k = \left\lfloor \frac{\text{Number of SNPs}}{\text{Number of SNPs per subdomain}} \right\rfloor$$

where k is the number of clusters. This results in the following amount of clusters:

Table 7.6: Conversion to number of clusters

Number of SNPs per subdomain	Clusters	Subdomains
Chromosome	5	6
200	61	62
100	123	124
50	246	247
10	1234	1235
3	4115	4116

Let Z be the SNP genotype matrix. Each column of this matrix represents a SNP and each row represents an animal. This matrix only contains the value $0, 1$ and 2 as described in chapter 6. The covariance matrix is defined as:

$$COV = Z^T Z$$

In other words, the covariance between SNPs is the dot product between each columns of Z. Therefore, a small or large covariance value does not necessarily mean that SNPs are unrelated or related, respectively. This could also be a reason why grouping per chromosome is not effective. Define the variance matrix as the matrix that only contains the main diagonal entries of the covariance matrix.

$$VAR = diag(Z^T Z)$$

Then the correlation matrix of SNPs can be defined as:

$$COR = VAR^{-\frac{1}{2}} \cdot COV \cdot VAR^{-\frac{1}{2}}$$

In the next paragraphs the k-means clustering method will be applied on the correlation matrix. Clustering of SNPs can also be applied on the SNP genotype matrix Z. Note that the matrix contains categorical data, because the genotypes are coded as $0, 1$ and 2. The k-means algorithm uses the Euclidean distance to create clusters. This distance measure is not suited for categorical data. Thus a different clustering method is recommended.
The correlation matrix of SNPs seems to be similar to the block matrix Σ^{33} in the coefficient matrix of the ssSNPBLUP. In the block matrix this term appears $\frac{1}{w} Z^T A_{gg}^{-1} Z$, also this block matrix corresponds to the SNP effects. Moreover, in the ssGBLUP model the correlation matrix appears in the genomic relationship matrix $G_{gg} = Z \frac{1-w}{k} I_{m,m} Z^T + wA_{gg}$.

Results of k-means

The clusters obtained from the k-means algorithm have the following size and ratios, only a few will be shown, see Appendix 10.3 for more figures:

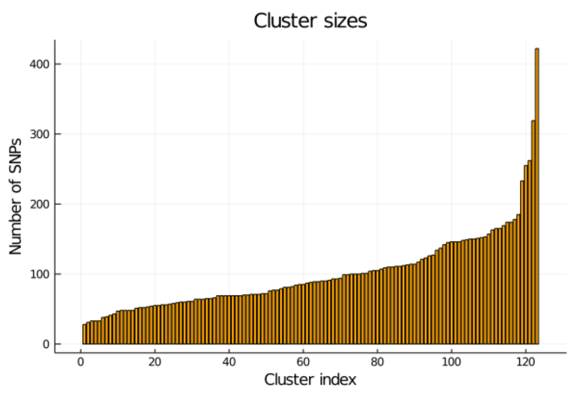


Figure 7.13: All cluster sizes with 123 clusters.

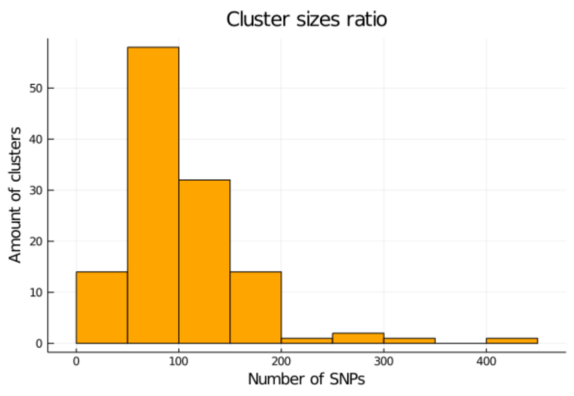


Figure 7.14: Histogram of the cluster sizes showing the ratio of the cluster sizes.

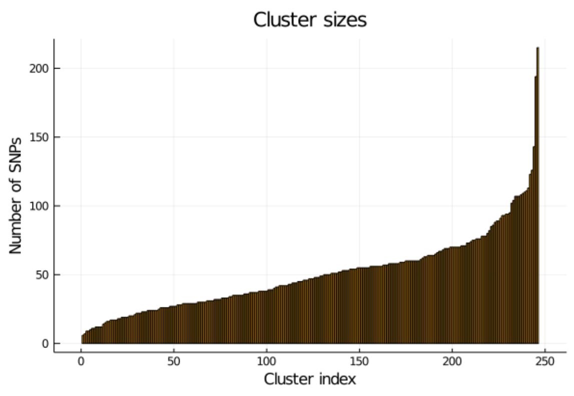


Figure 7.15: All cluster sizes with 246 clusters.

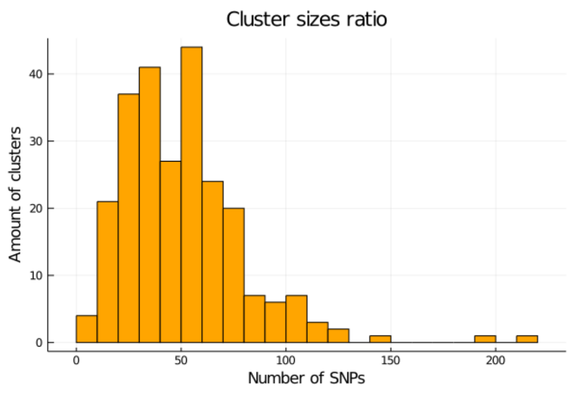


Figure 7.16: Histogram of the cluster sizes showing the ratio of the cluster sizes.

Noticeable that there are a few outliers, which are always the large cluster sizes. Most of the cluster sizes are centred around a certain value. This value is similar to the amount of number of SNPs per subdomain, the conversion can be seen in table 7.6. Heatmaps of the correlation matrix are shown in the next figures:

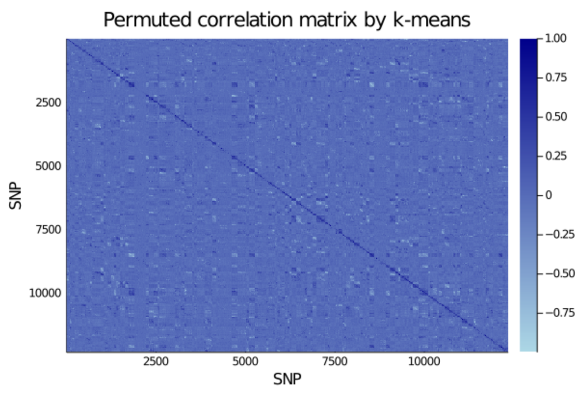


Figure 7.17: Correlation of the SNPs with 123 clusters sorted by the k-means algorithm.

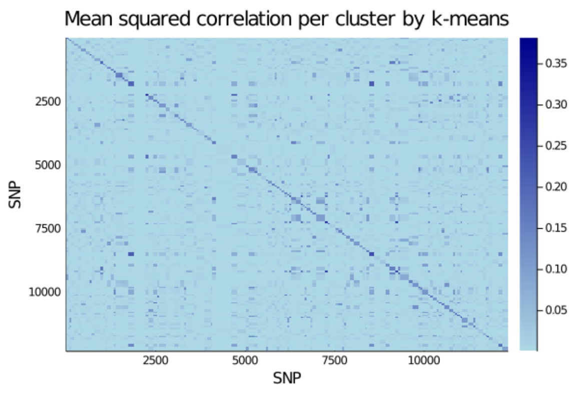


Figure 7.18: The average value of each cluster sorted by the k-means algorithm.

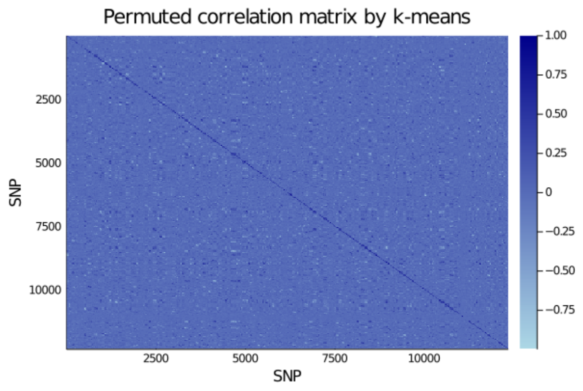


Figure 7.19: Correlation of the SNPs with 246 clusters sorted by the k-means algorithm.

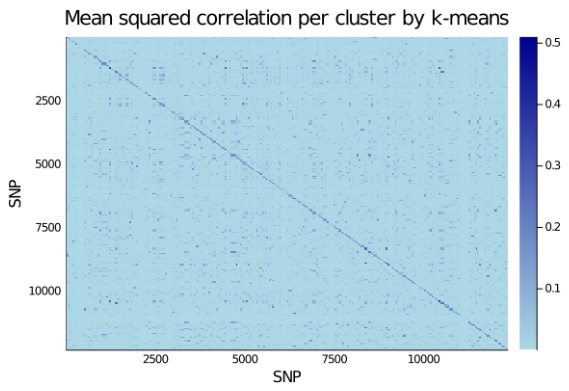


Figure 7.20: The average value of each cluster sorted by the k-means algorithm.

There is a mixture of relative low and high mean squared correlation of the clusters, which can be seen on the diagonals in figures 7.18 and 7.20. If more clusters are introduced, the pattern becomes finer. The results of the DPCG method are shown in the next figures.

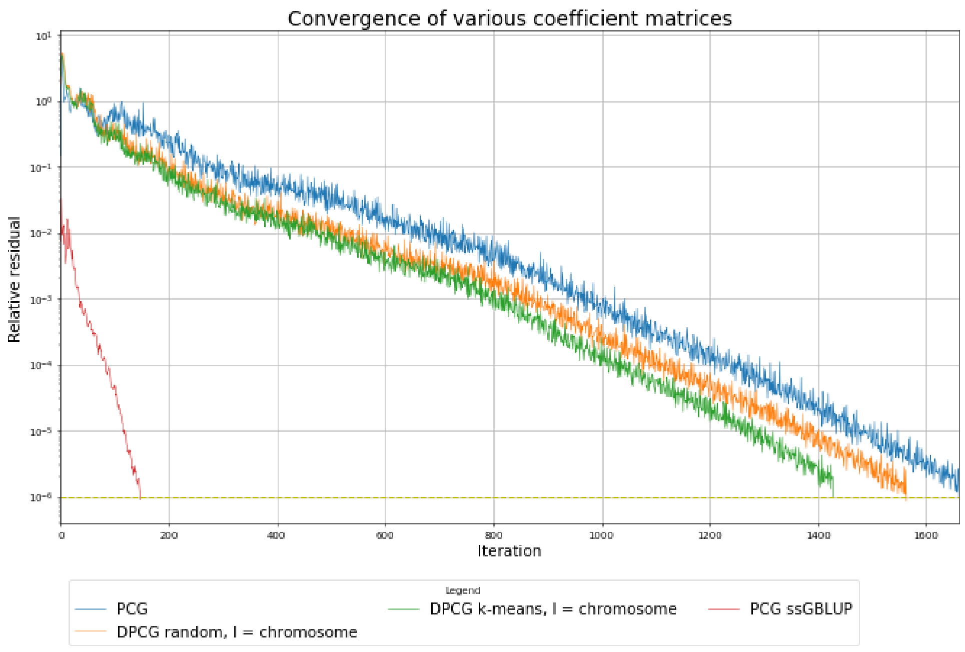


Figure 7.21: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with 5 clusters.

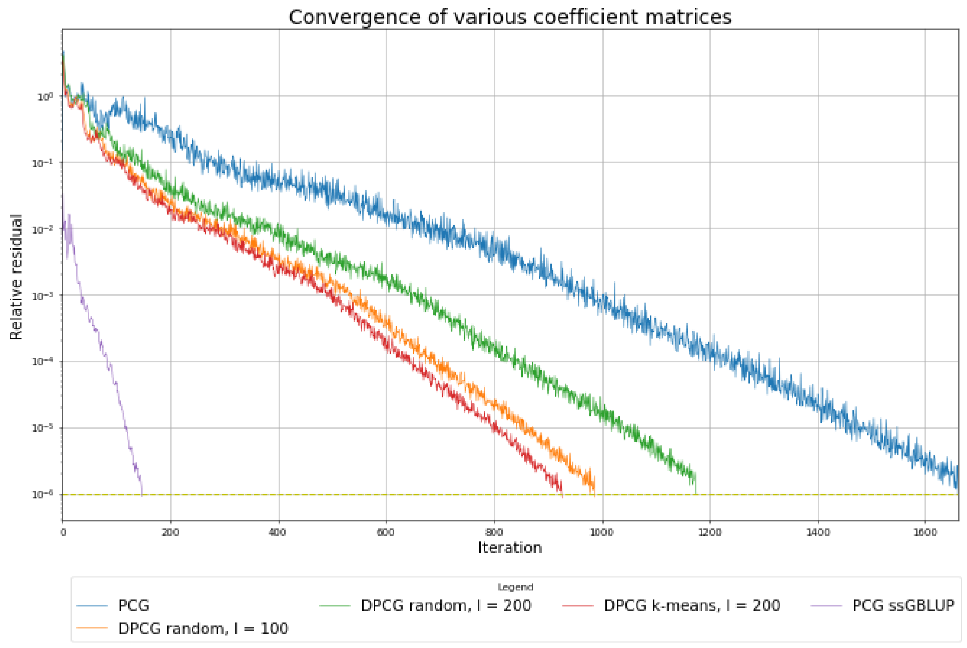


Figure 7.22: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with 62 clusters.

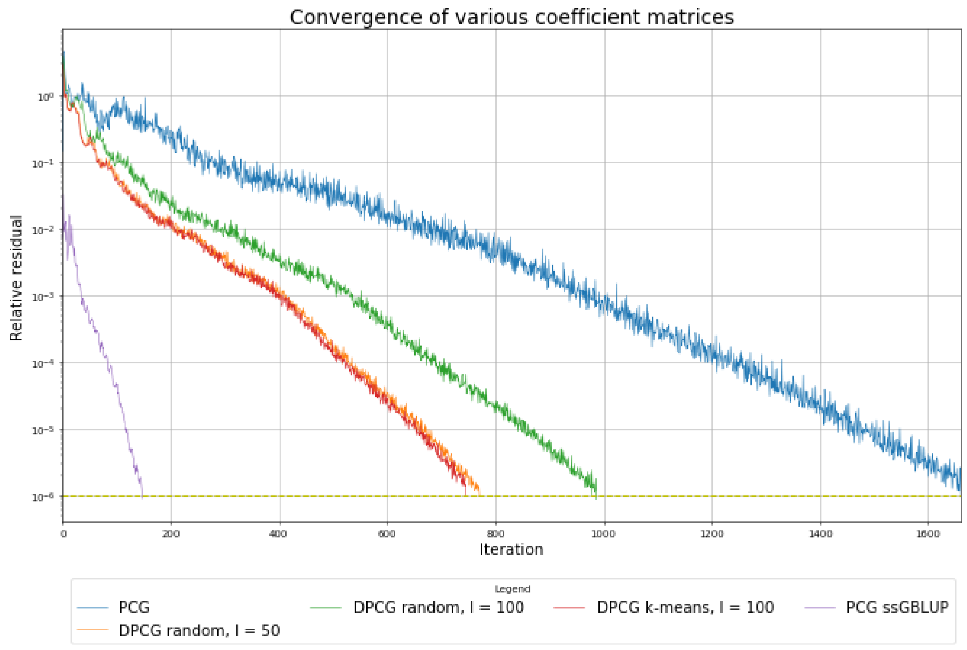


Figure 7.23: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with 123 clusters.

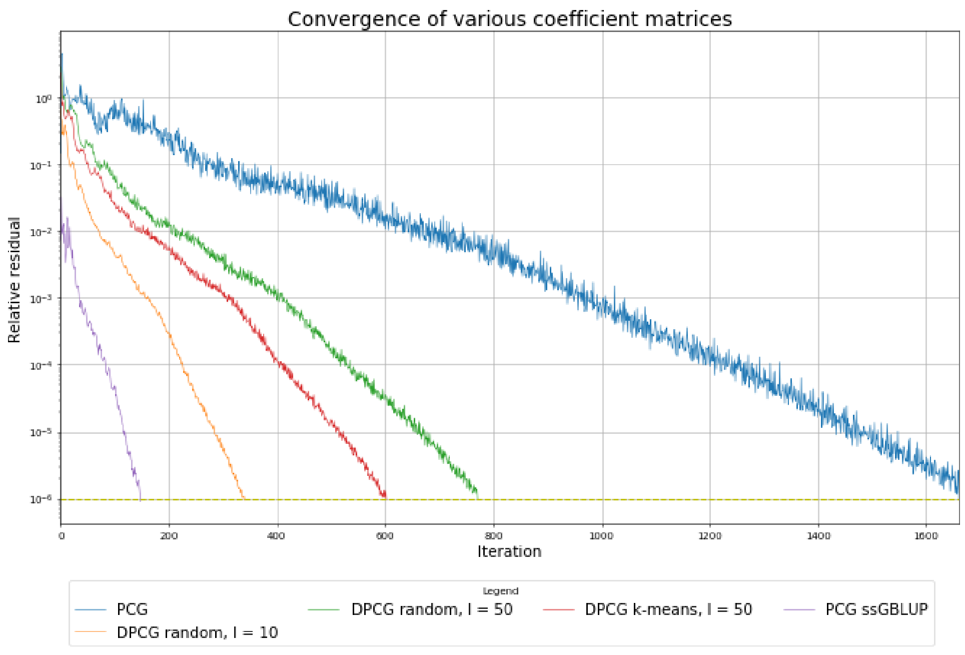


Figure 7.24: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with 246 clusters.

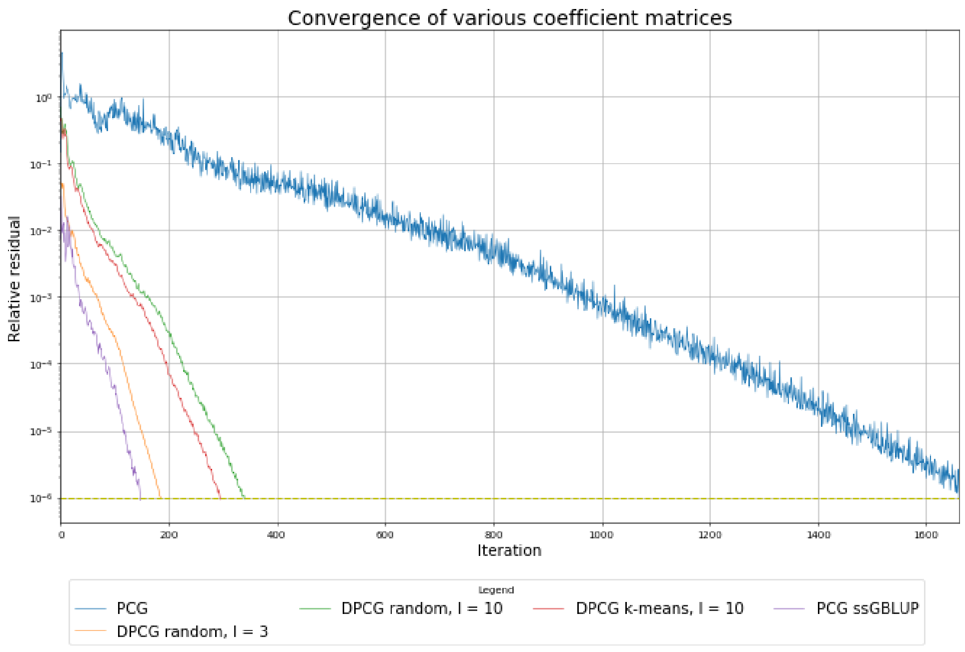


Figure 7.25: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with 1234 clusters.

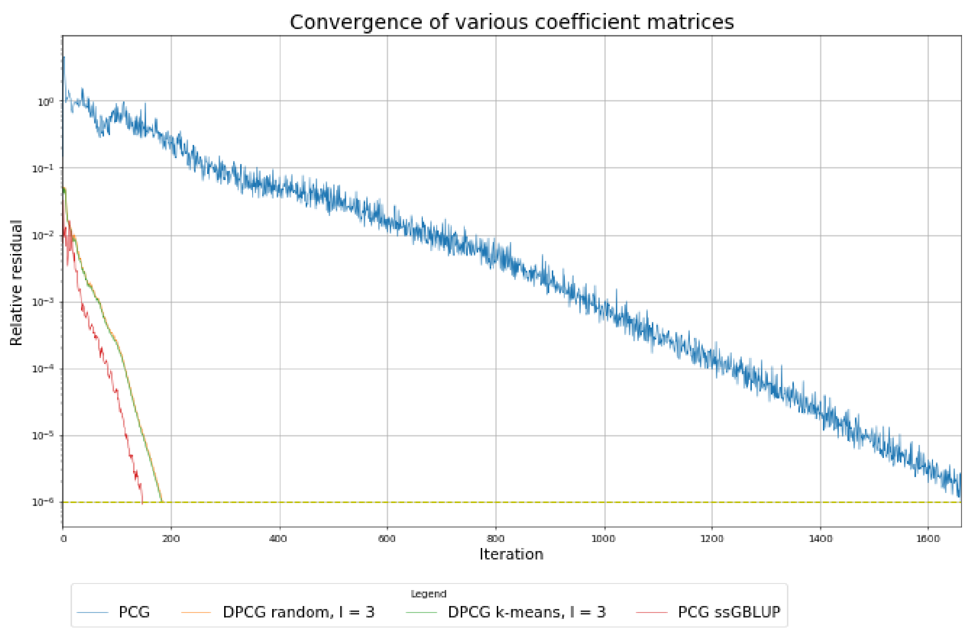


Figure 7.26: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with 4115 clusters.

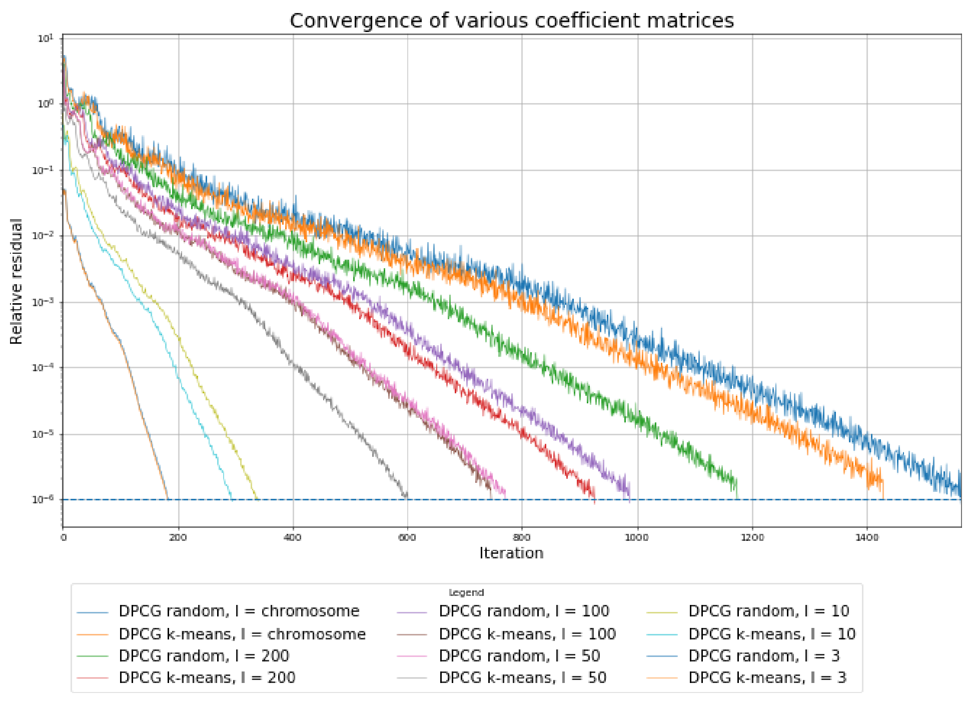


Figure 7.27: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm.

Table 7.7: The smallest and largest Ritz values of the deflated preconditioned matrix. Table 7.6 can be used to convert l to the amount of clusters for the k-means algorithm. The number of subdomains should be similar with the previous methods.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG	0.001435	323.05	226199.1	1662
DPCG random chromosome	0.001497	266.50	177994.46	1565
DPCG k-means chromosome	0.001497	203.64	136013.2	1430
DPCG random $l = 200$	0.001497	114.59	76529.6	1176
DPCG k-means $l = 200$	0.001497	70.74	47242.7	928
DPCG random $l = 100$	0.001497	78.02	52102.9	988
DPCG k-means $l = 100$	0.001498	38.93	25989.8	747
DPCG random $l = 50$	0.001498	41.46	27685.7	772
DPCG k-means $l = 50$	0.001498	24.20	16149.7	603
DPCG random $l = 10$	0.001500	7.11	4738.3	341
DPCG k-means $l = 10$	0.001502	5.49	3654.4	296
DPCG random $l = 3$	0.001505	2.30	1526.1	186
DPCG k-means $l = 3$	0.001509	2.30	1522.3	184
ssGBLUP	0.001799	2.47	1370.7	149

In figures 7.21-7.26, the results of applying the k-means algorithm to construct subdomains is plotted against results of random sampling without chromosome restriction. In other words, subdomains based on correlation are compared with subdomains chosen at random. An improvement in the convergence rate can be seen for all sizes. Applying the k-means algorithm with $l_{k-means} = \{100, 200\}$ yields a slightly better convergence than applying random sampling for $l_{random} = \{50, 100\}$. Thus halve the amount of subdomains is required for a similar convergence by applying the k-means algorithm for those cases. It seems that for a large amount of clusters the k-means algorithm is less effective in improving the convergence. This is due to the fact that the columns of the deflation-subspace matrix approximate the problematic eigenvalues quite well for small subdomain sizes. Note that for a larger amount of clusters yields smaller subdomain sizes. The cluster sizes for $l = \{100, 3\}$ can be seen in figures .
The improvement can be seen in table 7.8. The improvement is calculated by

$$\frac{\text{Number of iterations method } A - \text{Number of iterations method } B}{\text{Number of iterations method } A} \cdot 100\%$$

where method A refers to the method used in chapter 7.4.5 and method B is the k-means algorithm.

Table 7.8: Improvement compared with number of clusters

Subdomains	**Improvement**
6	8.62%
62	21.09%
124	**24.39%**
247	21.89%
1235	13.20%
4116	1.08%

The largest improvement is seen at 123 clusters. It seems that there are diminishing returns for small and large numbers of clusters. This is due to the fact that the subdomain sizes are too small or too large to make the k-means algorithm effective. For small subdomains, it is more likely that the columns of the deflation subspace matrix approximate the eigenvectors corresponding to the problematic eigenvalues better. For large subdomains, there are less columns, thus it is more likely that the span does not contain many eigenvectors corresponding to the problematic eigenvalues.

The improvement of convergence comes along with additional computation time from the k-means algorithm, which can be seen in table 7.9. The computation time for the deflation matrix includes the assembly of the deflation matrix, the k-means algorithm and the correlation matrix that is required for the algorithm. The column with header **DPCG** only accounts for the iteration process. The last column combines constructing the deflation matrix and the iterative solver computation times. The assembly of the ssSNPBLUP can be disregarded, since the same linear system is used for these results. The computations shown in table 7.9 and 7.10 are estimates, since background processes can influence the computation time.
It can be seen in table 7.9, when the number of clusters increases, the computation of the k-means algorithm is not monotonically decreasing. For a large amount of clusters the algorithm decreases in computation time. There are relatively less data points to be added to the clusters surrounding the cluster centres, for a large amount of clusters compared with the amount of data points. In the largest case 4115 cluster centres are chosen, which is about $\frac{4115}{12346} \cdot 100\% = 33.3\%$ of all data points.

Table 7.9: Computation times of the DPCG method including the k-means algorithm with various amount of clusters. Time is shown in **seconds**. The column with the name **Total** is the sum of **Deflation Matrix** and **Iteration**. The row in bold has the quickest total computation.

Clusters	***k*-means**	**Deflation Matrix**	**Iteration**	**Total**
5	10.35	27.58	2414.16	2441.74
61	106.61	123.55	1557.69	1806.72
123	231.93	249.03	1264.83	1513.86
246	244.67	262.09	1048.55	1310.64
1234	**577.82**	**595.94**	**597.59**	**1193.53**
4115	221.84	244.05	1190.39	1434.44

Table 7.10: Computation times of the DPCG method by random sampling the SNPs without the chromosome restriction. Time is shown in **seconds**. The row in bold has the quickest total computation time.

Subdomain size	**Subdomains**	**Deflation Matrix**	**Iteration**	**Total**
Chromosome	6	2.22	2836.99	2839.21
200	63	1.20	1951.52	1952.72
100	125	1.74	1697.50	1699.24
50	248	1.76	1366.59	1368.35
10	**1236**	**1.80**	**689.21**	**691.01**
3	4117	1.58	890.78	892.36

Table 7.11: Comparison of the computation times between table 7.9 and 7.10.

Random sampling size	***k*-means clusters**	**Deflation Matrix**	**DPCG**	**Total**
Chromosome	**5**	**-1142.34%**	**14.90%**	**14.00%**
200	61	-10195.83%	20.18%	7.48%
100	**123**	**-14212.07%**	**25.49%**	**10.91%**
50	246	-14791.48%	23.27%	4.22%
10	1234	-33007.78%	13.58%	-72.72%
3	4115	-15346.20%	-33.63%	-60.75%

Comparing table 7.9, 7.10 and 7.11, applying the k-means algorithm is faster for the cases $\{5, 61, 123, 246\}$, this corresponds to subdomain sizes $l = \{50, 100, 200, Chromosome\}$. In practice, the k-means algorithm is only run once for multiple genetic evaluations, since for those evaluations the same set of SNPs is used. Therefore, the k-means algorithm computation time is less important if multiple evaluations are needed.
Note that the assembly of the deflation matrix have similar computations by disregarding the overhead time of the k-means algorithm and calculation of the correlation matrix (which is excluded from these tables).
Observing the **DPCG** column, the k-means algorithm contains the quickest computation time of both subdomain selection strategies for 1234 number of clusters. Since it takes significantly less iterations for convergence, as can be seen in table 7.8.
The overhead of the k-means algorithm is relatively large for ≥ 1234 clusters, which makes the total computation longer than the random sampling method with subdomain sizes $l = \{3, 10\}$.
A trend can be seen when more subdomains are used, the absolute computation time decreases. Remarkably, for both methodologies the computation time increases from $l = 10$ to $l = 3$. Moreover, the DPCG method with the k-means clustering is slower, which is the opposite trend from the other cluster sizes. Thus k-means clustering with a lot of clusters seems to be less effective than random sampling with small subdomain sizes.
Comparing the total computation time improvement in 7.11, the largest relative decrease in total computation is seen for 5 clusters. By disregarding the overhead, the **DPCG** column shows that the largest improvement in computation time holds for 123 clusters. Although, 246 clusters has a similar improvement rate.
In general, the computation takes longer for a small number of clusters, as can be seen

by comparing the computation times of 5 clusters with 1234 clusters, see table 7.9. All in all, in absolute terms the best improvement is with 1234 clusters and in relative terms, disregarding the overhead of the k-means algorithm, the best improvement is with 123 clusters.

Results of equidistant and k-means++

The performance of the k-means algorithm relies on the initial cluster centres that are chosen. Therefore, two additional initial cluster centres selection strategies have been chosen for the following results:

1. The cluster centres are chosen by SNPs based on their index. It starts from the first index and steps of $\frac{\text{Number of SNPs - 1}}{\text{number of clusters}-1}$ are taken for the next index. When the addition with this step size returns a decimal number, then that number will be rounded down to the nearest integer. Thus the intial centres follow the map of the chromosomes with that given step size.

2. The cluster centres are chosen by applying the k-means++ algorithm.

Only results with 123 clusters will be discussed in this chapter, which amounts to a similar amount of subdomains by defining subdomains containing 100 SNPs. The cluster sizes follow a similar pattern as the first result, see figures 7.14, 7.28 and 7.30. Results with different number of clusters can be found in Appendix 10.3.

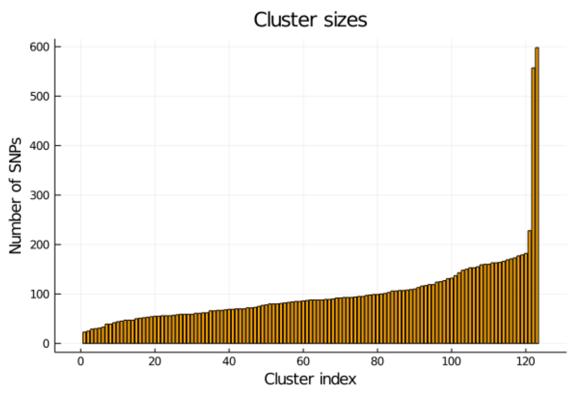


Figure 7.28: All cluster sizes with 123 clusters by taking the **initial clusters equidistantly**.

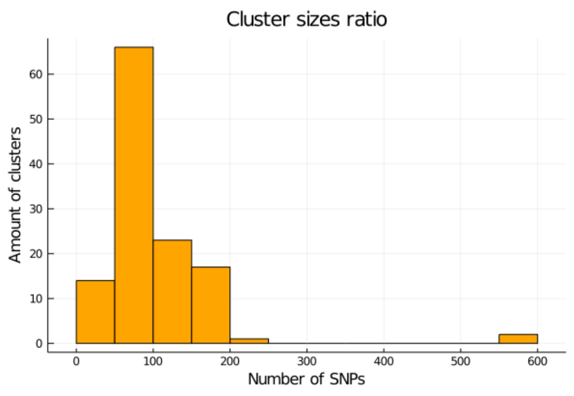


Figure 7.29: Histogram of the cluster sizes showing the ratio of the cluster sizes.

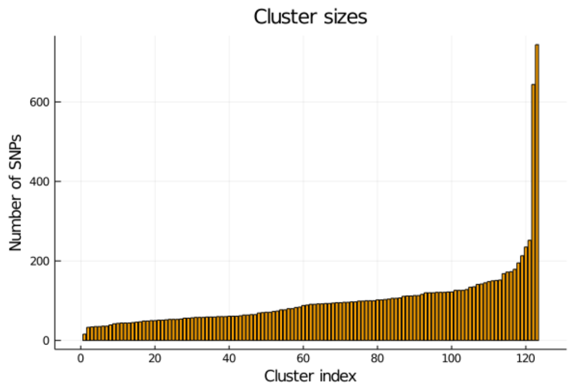


Figure 7.30: All cluster sizes with 123 clusters by applying the *k***-means++ algorithm**.

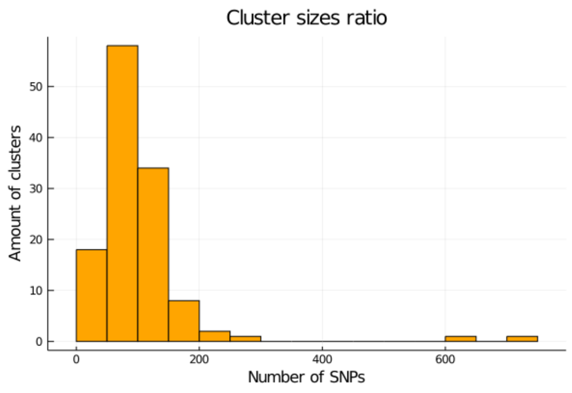


Figure 7.31: Histogram of the cluster sizes showing the ratio of the cluster sizes.

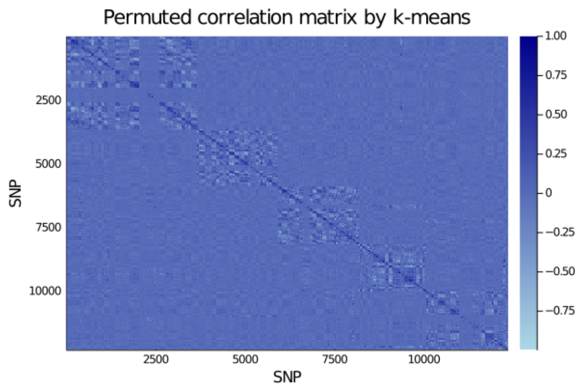


Figure 7.32: Correlation of the SNPs with 123 clusters sorted by the *k***-means algorithm using equidistantly chosen initial centres**.

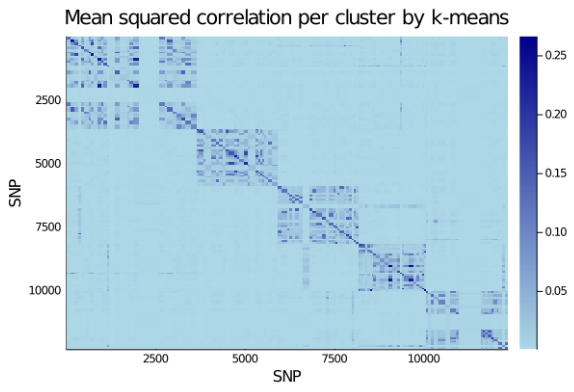


Figure 7.33: The average value of each cluster sorted by the *k*-means algorithm using equidistantly chosen initial centres.

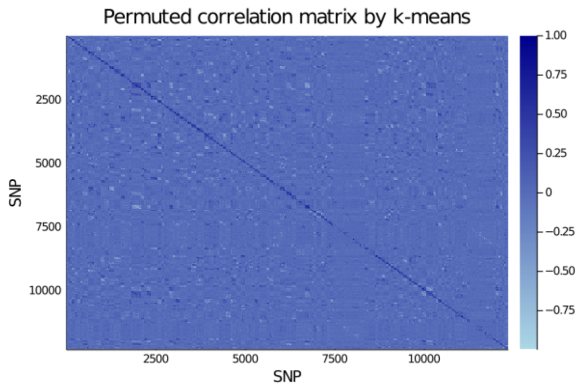


Figure 7.34: Correlation of the SNPs with 123 clusters sorted by the *k***-means++ algorithm**.

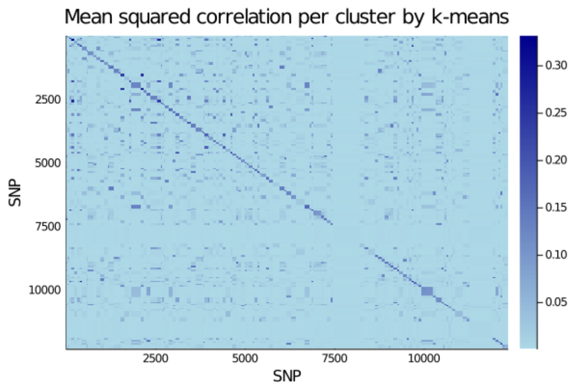


Figure 7.35: The average value of each cluster sorted by the *k*-means algorithm.

The clustering of choosing the initial centres equidistantly leads to a whole different correlation pattern, then the other methods as seen in figures 7.18, 7.33 and 7.35. Visually it can be generalised to five large clusters, this amount corresponds with the amount of chromosomes used in this data set. The convergence of both methods are compared with the random initial cluster centres method in figures 7.36 and 7.37.

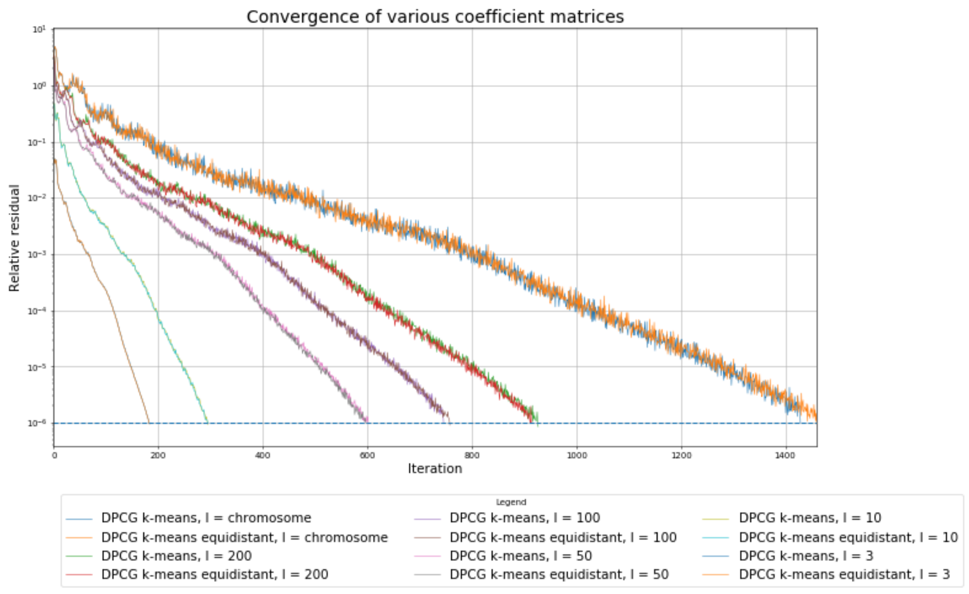


Figure 7.36: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means algorithm with the initial centres taken equidistantly.

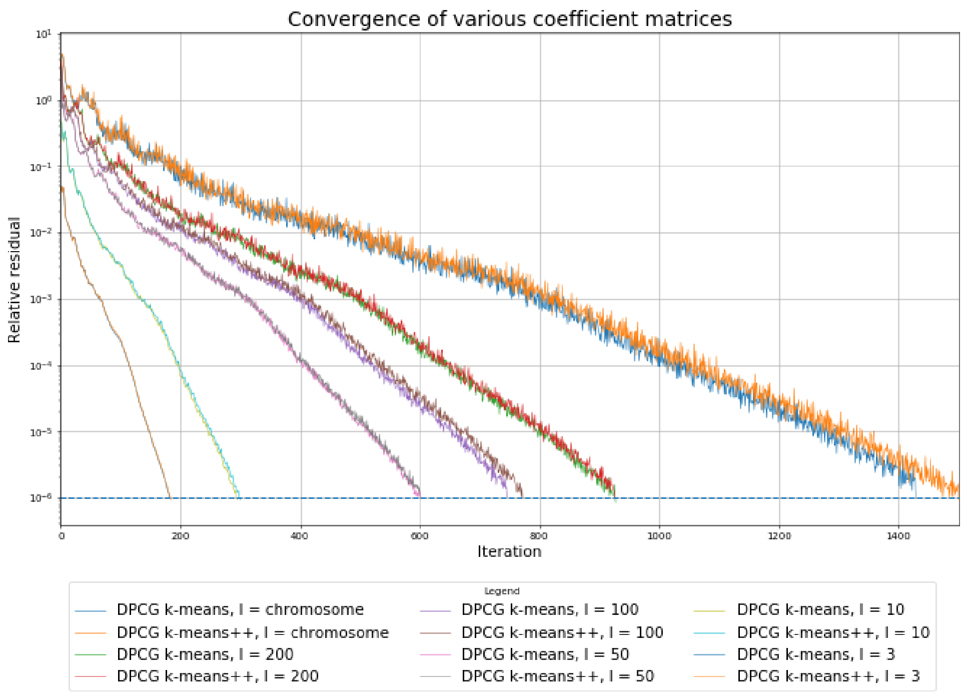


Figure 7.37: The convergence of the DPCG method where the subdomains are based on the correlation through the k-means++ algorithm.

Table 7.12: The smallest and largest Ritz values of the deflated preconditioned matrix. Table 7.6 can be used to convert l to the amount of clusters for the k-means algorithm. The number of subdomains should be similar with the previous methods.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
DPCG chromosome k-means equidistant	0.001497	217.73	145424.2	1461
DPCG $l = 200$ k-means equidistant	0.001498	61.17	40846.2	915
DPCG $l = 100$ k-means equidistant	0.001498	40.19	26832.3	759
DPCG $l = 50$ k-means equidistant	0.001498	24.09	16070.6	599
DPCG $l = 10$ k-means equidistant	0.001502	5.45	3631.1	296
DPCG $l = 3$ k-means equidistant	0.001510	2.30	1522.3	184

Table 7.13: The smallest and largest Ritz values of the deflated preconditioned matrix. Table 7.6 can be used to convert l to the amount of clusters for the k-means++ algorithm. The number of subdomains should be similar with the previous methods.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
DPCG chromosome k-means++	0.001497	228.14	152375.1	1502
DPCG $l = 200$ k-means++	0.001497	79.66	53196.9	927
DPCG $l = 100$ k-means++	0.001498	48.41	32321.6	773
DPCG $l = 50$ k-means++	0.001498	24.08	16070.6	602
DPCG $l = 10$ k-means++	0.001503	5.74	3818.0	301
DPCG $l = 3$ k-means++	0.001510	2.30	1521.8	185

The initial centres chosen equidistantly performs better than the k-means++ algorithm. Although, it does not perform better for all number of clusters compared with the initial cluster centres chosen at random. This seems to hold for $l = \{100, chromosome\}$, see figure 7.36. The k-means++ algorithm seems to converge 1 iteration earlier for $l = \{50, 200\}$, which is not a signifcant improvement. For all other cases it is slower than taking the initial cluster centres at random. However, in both cases the initial cluster centres are taken at random and the convergence can vary by a bit with a different seed.

In all cases, the k-means algorithm with different kinds of initial cluster centres reduces the number of iterations needed for convergence in similar trends.
The additional computation time of the k-means algorithm can be overlooked, if multiple genetic evaluations are done with the same SNPs. Also the computation time never exceeds the computation time of the DPCG method as seen in the results from table 7.9. Ideally, the subdomains are constructed such that they approximate the eigenvectors that correspond with the problematic eigenvalues. Applying the k-means algorithm seems to eliminate the large eigenvalues effectively. The clustering algorithm creates group of SNPs that can be related with one and another. Thus a group of SNPs is related to these large eigenvalues.

7.5 Concluding remarks

With these results, the research question can be answered. **It does matter how subdomains are chosen.** This can be seen by the application of the k-means algorithm. Moreover, the methods that apply random sampling SNPs per chromosome or random sampling SNPs without chromosome restriction or following the map do not improve the convergence at all or in a significant way.
From figures 7.14, 7.29 and 7.31 outliers of large clusters can be seen. Combining these large clusters could be interesting. This will result in a reduction of 1-2 subdomains, which probably will not improve the computation significantly. Moreover, there always seem to be a few clusters that are relatively large in size compared with the other clusters for different numbers of cluster, this trend can be seen in the histograms in chapter 7.4.6 and Appendix 10.3. Therefore, a second clustering can be formed with the large clusters, to see if any new clusters are formed that could affect the convergence in a beneficial way. This can also be done with all small clusters combined. Doing so could increase the amount of clusters. Consequently, it will increase the amount of subdomains and that makes the DPCG method more expensive.
Another approach would be applying the k-means algorithm on each individual chromosome or groups of chromosome. Although, the results from random sampling with and without chromosome restriction, conclude that sampling for each chromosome does not improve the convergence.
Determining the optimal number of clusters that is required, such that the clusters are constructed in a beneficial way for the convergence rate of the DPCG method. The overall computation time of the whole process should also be taken into account.
The k-means algorithm clusters SNPs based on similarity of the given initial centres. This algorithm has been applied on only one case. Thus this case could be a lucky case, where the initial centres are chosen such that the k-means algorithm converges to a global minimum instead of a local minimum. Hence, research can be extended on different systems of linear equations resulting from the ssSNPBLUP model.
It can be interesting to apply different clustering methods on the SNP genotype matrix Z. In a broader sense, a machine learning technique or pattern recognition method could be investigated. Note that the matrix Z can be classified as categorical data, since the genotypes are categorised by 3 values. Thus caution should be taken on choosing the proper method.

CHAPTER 8

POD Based Deflation

8.1 Data

The same data described in chapter 7.1 will be used from this point on. In genetic evaluation new data is added from on a routinely basis. By incorporating this data in the model, the linear system will grow in dimensions. This process of adding new data will be referred as updating in this thesis. For updating, the current data will be reduced to simulate this process. The reduction of data is between $0\%-3\%$. In practice, the updated system usually grows with a rate lower than 1%. Hence multiple updates are simulated. The reduction will be applied on animals with phenotype records and genotype records. Table 8.1 shows the amount of records up to a reduction of 3% to illustrate what the reduction entails.

Table 8.1: Amount of phenotype and genotype records after reduction.

Reduction	Phenotype	Genotype
0%	82118	18678
0.25%	81912	18631
0.5%	81707	18584
0.75%	81502	18537
1%	81296	18491
2%	80475	18304
3%	79654	18117

In chapter 8.2 a specific choice of an initial solution for the updated system will be analysed. This initial solution will be the solution of the previous genetic evaluation. However, this solution does not match the dimensions of the updated system, since it does not contain entries corresponding to the new animals. Note that the solution can be divided in 3 parts: fixed effects, estimated breeding values (EBV) and SNP effect. The missing entries are the EBVs. Assuming that 50% of DNA is inherited from each parent, a straightforward approximation of these missing entries is by taking the average of the

EBVs from the parents of the corresponding animal.

The approximation can be obtained by solving the reduced linear system with the reduced pedigree matrix, where the reduced pedigree matrix is the pedigree without the updated animals. After solving this system, the updated pedigree matrix can be used to search the parents of the updated animals, then the average of the EBVs can be taken. Thereafter, the approximated values are combined with the solution from the previous genetic evaluation.

Assume that the updated pedigree is known beforehand, then there is an alternative way to obtain this approximation. This can be done by solving the reduced linear system, where the updated pedigree matrix has been incorporated instead of the reduced pedigree matrix. Thus there is no extra step involved to search the parents, since this is already incorporated in the reduced system with the updated pedigree matrix. Consider equation (6.22), the reduction affects the following matrices:

- the incidence matrices related to fixed effects $X \in \mathbb{R}^{n_T \times n_F}$
- the incidence matrix related to additive genetic effects $W \in \mathbb{R}^{n_T \times n}$
- the design matrix containing SNP genotypes $Z \in \mathbb{R}^{n_g \times m}$

The reduction applied on the incidence matrices X and W removes rows corresponding to updated animals. However, the assembly of Z should be taken with care, since the minor allele frequency is different for the reduced and updated animals. This is due to the changes in the ratio of animals with a given allele, since the population size is different and new animals with a different allele can be introduced. Therefore, different frequencies can be observed in the set of reduced and updated animals. Thus matrix Z is constructed with the updated genotyped animals, then a reduction is applied by removing the rows corresponding to the updated genotyped animals. This results in a system with the same dimension as the updated system. In the reduced system the updated animals are only accounted by the inverse of the (co)variance matrix of additive genetic effects of animals. Note that this matrix includes the inverse of the pedigree matrix, see equation (6.21). This reduced system calculates the EBVs of the updated animals by taking the average of the EBVs of their parents. The reason why this holds true is outside the scope of this research. This concept can be extended to the ssSNPBLUP model seen in equation (6.26).

8.2 Estimating EBV for reduced systems

A common practice in genetic evaluations when selecting an initial solution, is to take the solution from the previous evaluation. The initial solution is smaller in dimension, since it does not contain the estimated breeding values (EBV) of the new animals. These missing EBVs are estimated by taking the average of the EBVs of the parents from the corresponding animal.

The solutions of the reduced and original system are obtained with the DPCG method. A reduction has been applied up to 3% with steps of 0.25%. Results between 0.5 − 1% can be found in Appendix 10.6. The results between 1.25% − 2.75% are not reported in this thesis, because the results are quite similar. The solutions can be seen in figure 8.1-8.3. The deflation method of choice is random sampling of SNPs described in chapter 7.4.5. The subdomain sizes are $l = \{10, chromosome\}$. Subdomain size $l = 10$ yielded the quickest computation time of all methods shown in chapter 7.4. Similar results were obtained with $l = chromosome$, which are not shown in this thesis.

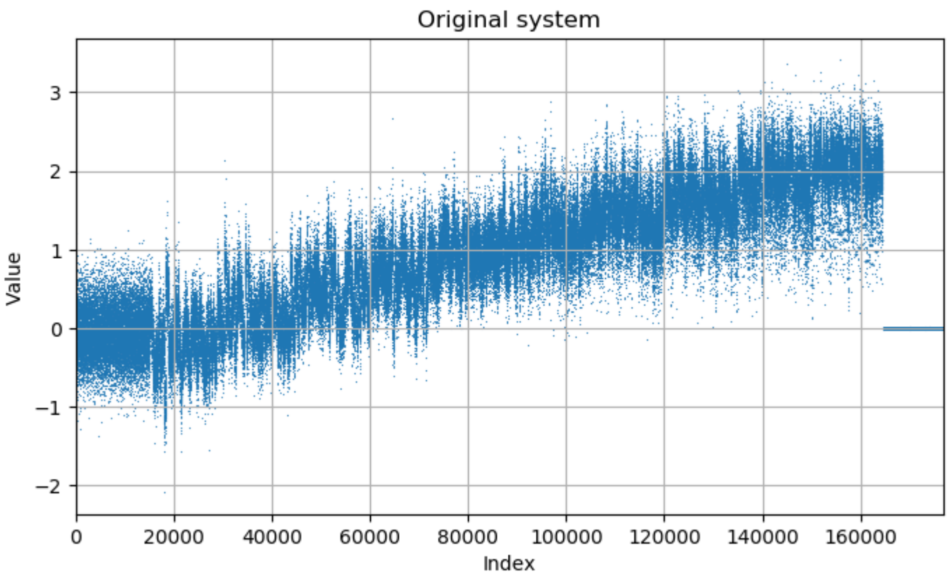


Figure 8.1: The solution of the original system.

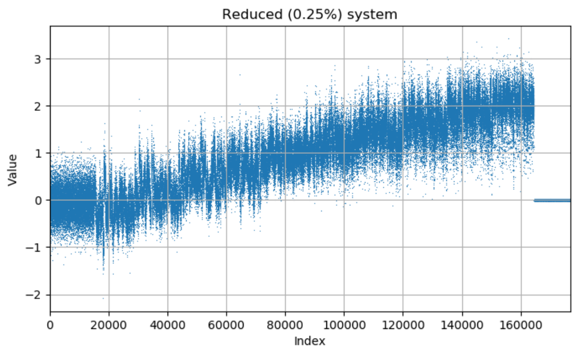


Figure 8.2: The solution of the reduced system with a reduction of 0.25%.

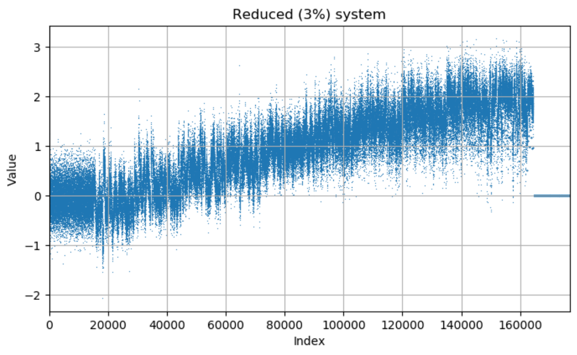


Figure 8.3: The solution of the reduced system with a reduction of 3%.

It is quite hard to see if there are differences between the solutions in figures 8.1-8.3. Therefore, a scatter plot with the original system on the x-axis and the reduced system on the y-axis, a plot of the absolute differences and the absolute of the relative difference should help visualise the differences as can be seen in figures 8.4-8.9.

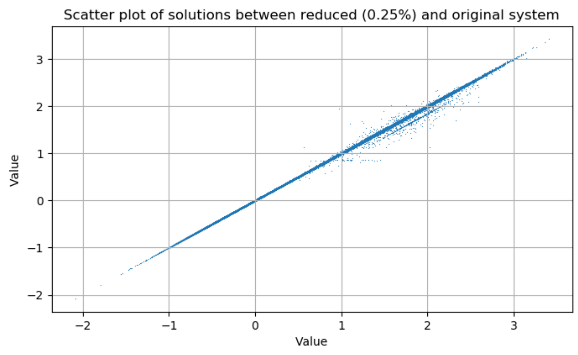


Figure 8.4: Comparison of values between solution of reduced and original system with a reduction of 0.25%.

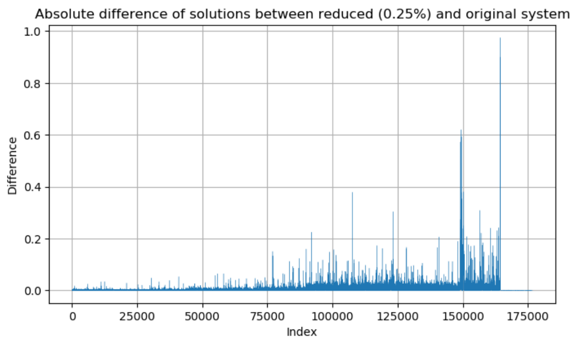


Figure 8.5: Absolute difference component wise of the solution of the reduced and original system with a reduction of 0.25%.

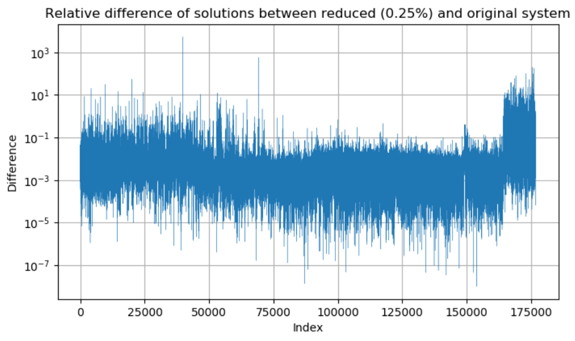


Figure 8.6: Relative difference component wise of the solution of the reduced and original system with a reduction of 0.25%.

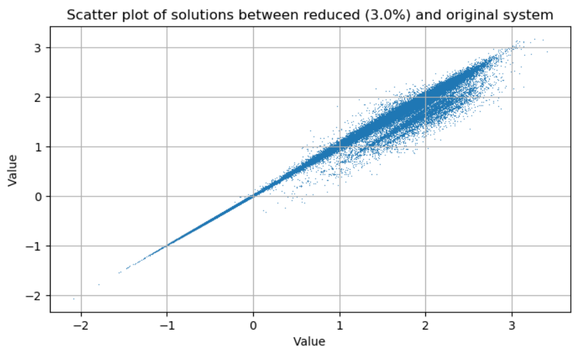


Figure 8.7: Comparison of values between solution of reduced and original system with a reduction of 3.0%.

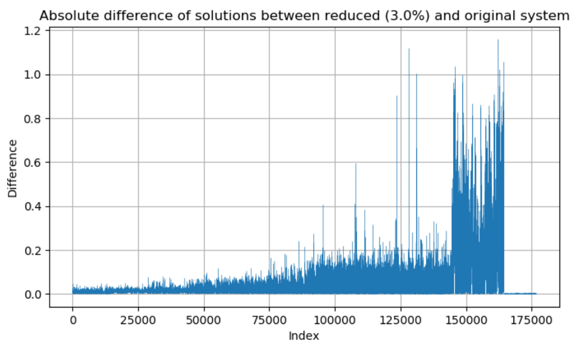


Figure 8.8: Absolute difference component wise of the solution of the reduced and original system with a reduction of 3.0%.

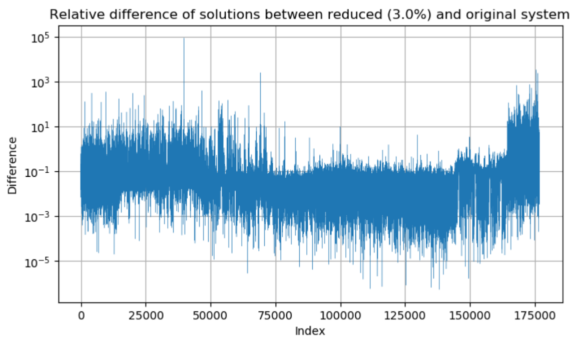


Figure 8.9: Relative difference component wise of the solution of the reduced and original system with a reduction of 3.0%.

The scatter plots can be interpreted as follows. A straight line with a regression coefficient 1 shows that the values match perfectly with each other. Small deviations from the straight line shows that the solutions are slightly different. In general the solutions of the reduced system resemble the solution of the original system. The resemblance is stronger when the reduction is smaller.
For the scatter plots, it can be observed that there is a second line parallel to the large line. The values of the original system are larger than the reduced system for this second line. This could be caused by new animals that are related with some of the old animals. Thus the new animals affect the values of the old animals.

Note that the animals are sorted from old to young. It can be seen that the absolute difference grows larger for newer generations as seen in figures 8.5 and 8.8. Thus adding new animals to the data will affect newer animals more than older animals in an absolute sense.
The SNP effect values which can be seen at the end of the tails in figures 8.5 and 8.8 do not seem to vary a lot in an absolute sense. Although, observing the relative difference, the largest differences on average correspond to the SNP effects. Moreover, the relative difference seems to be larger for older animals. Thus the relative difference seems to represent the opposite relations compared with the absolute difference.

The convergence behaviour of the DPCG method with subdomain sizes $l = 10$ and $l = chromosome$ by taking the solution of the reduced system with reductions 0.25%, 0.5%, 0.75% and 1% as the initial solution can be seen in figures 8.10 and 8.11.

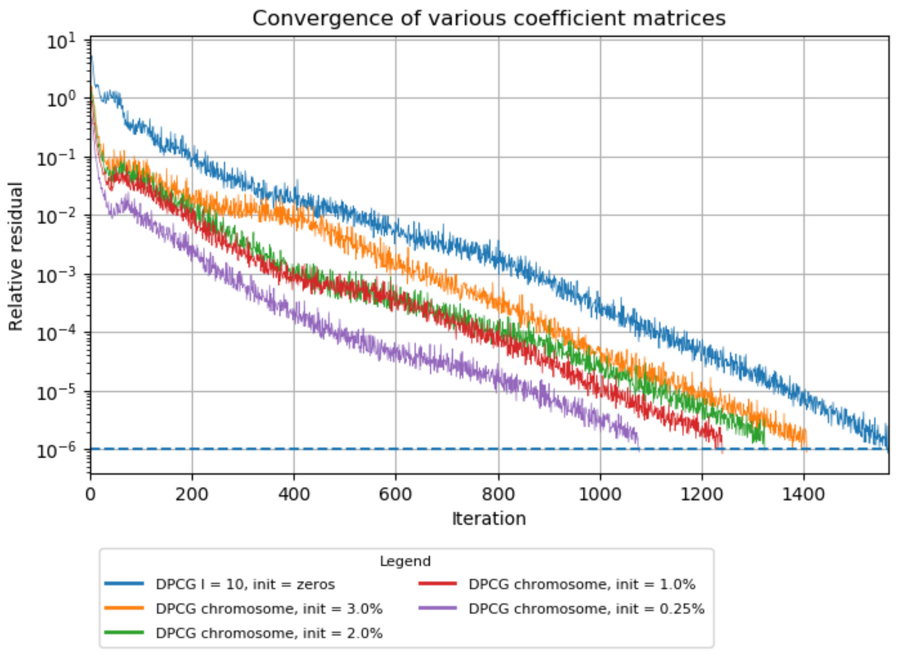


Figure 8.10: Convergence of the DPCG method where the inital solution is chosen as the solution from the reduced system. The subdomain sizes equals the chromosome sizes.

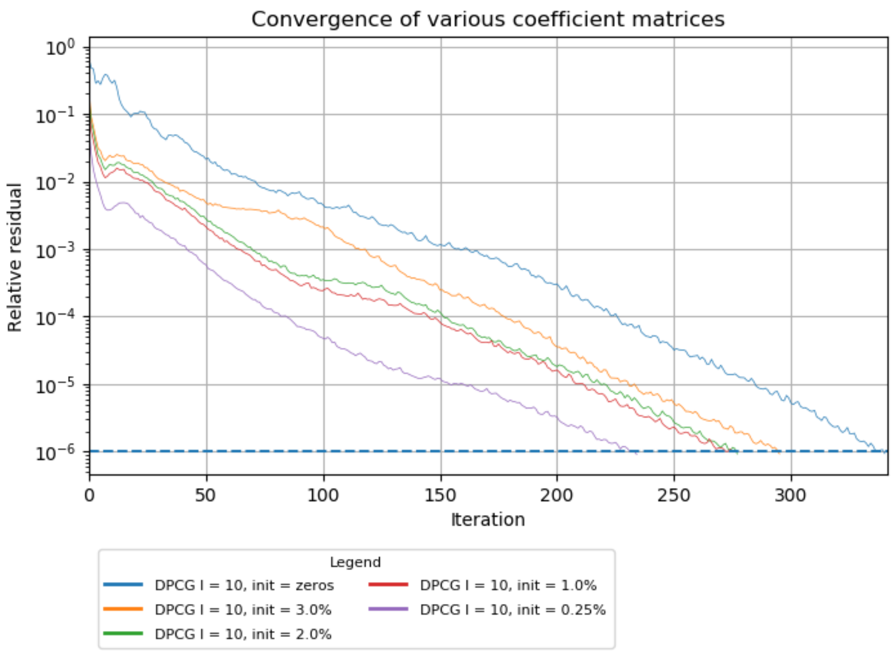


Figure 8.11: Convergence of the DPCG method where the inital solution is chosen as the solution from the reduced system. The subdomain size equals $l = 10$.

It does take less iterations to reach convergence when the initial solutions are approximated solutions of the original system, instead of the zero solution. The reduction in iterations is between 10% and 30%, which can be seen in figure 8.12.

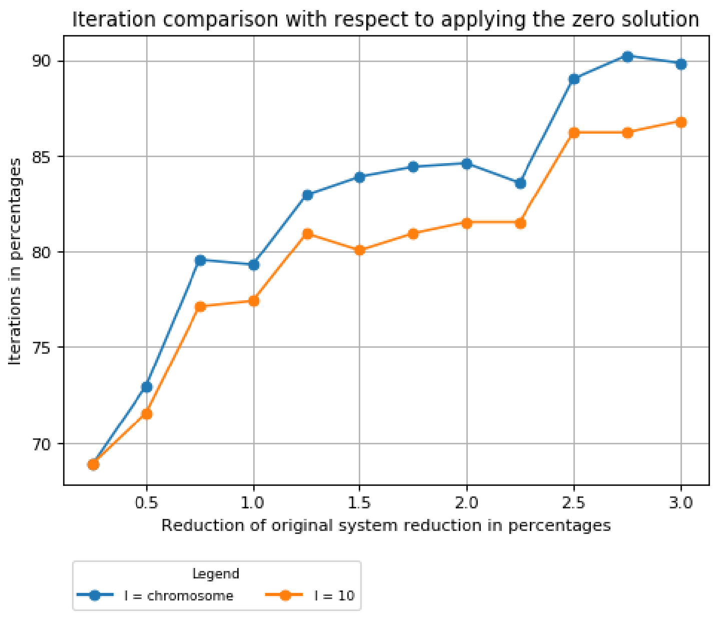


Figure 8.12: The amount of iterations needed for convergence by choosing the initial solution as the solution of the reduced original system.

An obvious trend is that the improvement is larger for a smaller reduction of the

original linear system. Since the solution of the reduced system with a smaller reduction will resemble the solution of the original system more, compare figure 8.4 and 8.7. Hence, the initial solution is closer to the actual solution and less iterations are required for convergence.
All in all, a change from the original linear system yields a change in the solution. The smaller the change, the smaller the change in the solution.

8.3 Proper Orthogonal Decomposition (POD) Deflation vectors

In this thesis the Proper Orthogonal Decomposition will be applied. This method creates deflation vectors based on the previous solutions. These deflation vectors are obtained by applying the Singular Value Decomposition (SVD) method on a data matrix. The POD based deflation method applied on a two-phase reservoir simulation, studied in [6], only required a few deflation vectors for an improvement in convergence. However the problem in this thesis is different, there is a possibility that a similar result can be obtained. Thus this could lead to use less deflation vectors compared with the subdomain decomposition studied in chapter 7.4. Hence, the POD method makes the deflation method cheaper compared to the subdomain decomposition method.
A brief explanation on the SVD method and POD method will be given. Thereafter, a description of the algorithm will be given and results of the updated ssSNPBLUP model will be discussed.

8.3.1 Singular Value Decomposition

This matrix decomposition is best described for real matrices, but it works for complex matrices as well. Let $A \in \mathbb{R}^{m\times n}$ with $m, n \in \mathbb{N}_{>0}$. Then there exists orthogonal matrices $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ such that

$$U^T AV = \Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_p) \text{ with } p = \min(m, n)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_p \geq 0$. The columns of U are called left singular vectors of A. The columns of V are called right singular vectors of A. The diagonal entries of Σ are the singular values of A, see ([7], page 76) for a proof of existence. The decomposition has the following properties for a matrix $A \in \mathbb{R}^{m\times n}$ with $m \geq n$:

- Let $U = \begin{bmatrix} u_1 & u_2 & \dots & u_m \end{bmatrix}$ and $V = \begin{bmatrix} v_1 & v_2 & \dots & v_n \end{bmatrix}$ where u_i and v_i are orthonormal vectors, respectively. Then the following holds:

$$A^T Av_i = \sigma_i^2 v_i$$
$$AA^T u_i = \sigma_i^2 u_i$$

  These equations follow directly from algebraic manipulation of the LHS. The equations show that there is a relationship between the SVD of $A^T A$ and AA^T. Equivalently, the following holds:

$$A^T A = V\Sigma^T\Sigma V^T$$
$$AA^T = U\Sigma\Sigma^T U^T$$

This is similar to the diagonalisation of A^TA and AA^T. Thus V corresponds to the eigenvectors of A^TA and U corresponds to the eigenvectors of AA^T.

$$\Sigma\Sigma^T = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m, \lambda_{m+1}, \ldots, \lambda_n)$$
$$\Sigma^T\Sigma = \text{diag}(\lambda_1, \lambda_2, \ldots \lambda_m)$$

where λ_i are the eigenvalues of A^TA and AA^T and $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$, since $\lambda_i = \sigma_i^2$. Note that A^TA and AA^T have m eigenvalues that are the same. $\lambda_i = 0$ for $i > m$, because the i-th column of Σ with $i > m$ are zero.

- If A has r positive singular values, then $rank(A) = r$ and

$$\mathcal{N}(A) = \text{span}\{v_{r+1}, v_{r+2}, \ldots, v_n\}$$
$$\mathcal{R}(A) = \text{span}\{u_1, u_2, \ldots, u_r\}$$

  Thus the columns of U corresponding to the largest singular values span the range of A. The columns of V corresponding to the smallest singular values span the nullspace of A. Moreover, these singular values are 0.

- If A has rank r, then there is an alternative way to formulate the matrix A:

$$A = \sum_{i=1}^{r} \sigma_i u_i v_i^T$$

  which is a sum of r rank-1 matrices.

Details on proofs of these properties can be found in ([7], chapter 2).

8.3.2 Proper Orthogonal Decomposition (POD)

One way to use the previous solutions, is to recycle the previous solutions in CG iterations. This can be done by applying a Proper Orthogonal Decomposition (POD) combined with the deflation method. In other words, the POD method creates vectors which are chosen as deflation vectors.

The POD method creates a small set of orthonormal basis vectors $\{\phi_1, \phi_2, \ldots, \phi_p\}$ with $\phi_i \in \mathbb{R}^n$ and $p \in \mathbb{N}$. This set can be seen as a projection of the original model in a lower dimension. The basis vectors ϕ_i are eigenvectors that usually correspond to the p largest eigenvalues of the data matrix:

$$R = \frac{1}{m-1} XX^T$$

where $X = \begin{bmatrix} x_1 & x_2 & \ldots & x_m \end{bmatrix} \in \mathbb{R}^{n\times m}$ with $x_i \in \mathbb{R}^n$ being a snapshot. The snapshots are chosen as the previous solutions as described in chapter 8.2.

The eigenvectors of R can be obtained by applying SVD on R. Define $X = U\Sigma V^T \in \mathbb{R}^{n\times m}$ where $U \in \mathbb{R}^{n\times n}, V \in \mathbb{R}^{m\times m}$ are orthogonal matrices containing the left singular vectors and right singular vectors, respectively. Moreover, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_m) \in \mathbb{R}^{n\times m}$ and $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_m \geq 0$. The following can be obtained:

$$R = \frac{1}{m-1}XX^T = \frac{1}{m-1}U\Sigma\Sigma^T U^T \in \mathbb{R}^{n\times n} \tag{8.1}$$

Recall that U corresponds with the eigenvectors of R, see chapter 8.3.1. Thus p eigenvectors can be chosen from U which corresponds to the largest eigenvalues.

In practice, the amount of snapshots is smaller than the amount of unknowns in a linear system. Consider the case $m << n$. Recall, that the eigenvalues of $\frac{1}{m-1}XX^T \in \mathbb{R}^{n\times n}$ and $\frac{1}{m-1}X^TX \in \mathbb{R}^{m\times m}$ contain the same eigenvalues. Note, that the latter matrix has a smaller dimension due to the assumption. Thus applying SVD on the matrix $\frac{1}{m-1}X^TX$ yields less computation to obtain the same eigenvalues. The following relation holds:

$$\frac{1}{m-1}X^TX = \frac{1}{m-1}V\Sigma^T\Sigma V^T$$

Hence, the right singular vectors V and the eigenvalues of R can be obtained from this SVD. Moreover, the eigenvectors of XX^T can be obtained from U, which can derived from $X = U\Sigma V^T$, the following holds:

$$U = XV\Sigma^{-1}$$

If Σ is a singular matrix or non square, the pseudoinverse of Σ can be used. This is a diagonal matrix containing the reciprocal of Σ and it takes value 0 if the diagonal entry of Σ is 0. It can be seen from equation (8.1) that the eigenvalues of R are the squared singular values. Thus the former equation can be formulated as:

$$U = XV\Lambda^{-\frac{1}{2}}$$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m) \in \mathbb{R}^{m\times m}$ with $\lambda_i = \sigma_i^2$ which are eigenvalues of X^TX.

Define $x \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Consider the linear system:

$$Ax = b$$

The POD basis consists of p orthonormal vectors corresponding to the p largest eigenvalues of R. The solution can be approximated as [6]:

$$x \approx \sum_{i=1}^{p} c_i\phi_i \tag{8.2}$$

with constants $c_i \in \mathbb{R}$.

The following approach of the POD method is not applied in this research, but could be interesting to keep in mind. The POD method can reduce the dimensions of the original system. Define a matrix $\Phi = \begin{bmatrix}\phi_1 & \phi_2 & \ldots, \phi_p\end{bmatrix}$, then the reduced system is given by:

$$\Phi^T A\Phi z = \Phi^T b$$

with $x = \Phi z$ and $z \in \mathbb{R}^p$. This can be seen as the linear system being projected onto the subspace spanned by the POD basis.

8.3.3 Deflation vectors

The POD basis vectors described in 8.3.2 are used as deflation vectors for the deflation-subspace matrix Z. Another motivation for this choice of the deflation vectors is due to the following lemma:

Lemma 8.3.1. *Consider the linear system*

$$Ax = b$$

with $A \in \mathbb{R}^{n\times n}$ an SPSD matrix, x, $b \in \mathbb{R}^n$ and b is chosen such that the linear system is consistent. Let the deflation subspace matrix Z be defined as

$$Z = \begin{bmatrix} x_1 & x_2 & \dots & x_p \end{bmatrix}$$

where $x = \sum_{i=1}^{p} c_i x_i$ with $x_i \in \mathbb{R}^n$ linear independent. Also the following identity holds:

$$x = Zc \tag{8.3}$$

with $c = \begin{pmatrix} c_1 & c_2 & \dots & c_p \end{pmatrix}^T$. Moreover, define the deflation matrix $P = I - AQ$ with $Q = ZE^{-1}Z^T$ and $E = Z^TAZ$. Define $\hat{x}$ as the solution of the system

$$PA\hat{x} = Pb \tag{8.4}$$

Then the solution of this linear system is obtained with one iteration of DCG if and only if $\hat{x} \in \mathcal{N}(P^T)$ or $\hat{x} = 0$.

Proof. "$\Rightarrow$". Assume that the DCG method has converged in one iteration. Thus x is a solution of $Ax = b$ and $\hat{x}$ is a solution of $PA\hat{x} = Pb$. Moreover, the following equality holds:

$$x = Qb + P^T\hat{x} \tag{8.5}$$

Note that Qb can be reformulated as:

$$Qb = QAx \overset{(8.3)}{=} QAZc = Zc = x$$

Thus substituting this into equation (8.5) yields:

$$P^T\hat{x} = 0$$

This can only be satisfied if $\hat{x} \in \mathcal{N}(P^T)$ or $\hat{x} = 0$.

"$\Leftarrow$". Assume that $\hat{x} \in \mathcal{N}(P^T)$ or $\hat{x} = 0$. The following holds:

$$Qb + P^T\hat{x} = QAx + 0 = QAZc = Zc = x$$

Also $Ax = b$ and $PA\hat{x} = Pb$ should be satisfied. The former is trivial, since it is assumed that x is the solution of $Ax = b$. The latter can be seen by rewriting $PA\hat{x} = Pb$ as:

$$\begin{aligned} &PA\hat{x} = Pb = PAx = PAZc = 0c = 0 \\ \iff &AP^T\hat{x} = 0 \\ \iff &0 = 0 \end{aligned}$$

Thus the DCG iteration converges in one iteration. □

In practice, obtaining these vectors is equivalent to solving this system. However, a close approximation of this deflation subspace could provide an improvement in the convergence of the DCG method. As seen in equation (8.2) the solutions can be approximated with a linear combination of the POD basis vectors. This result also holds for the preconditioned system, therefore an improvement in the DCG method would also hold for the DPCG method.

8.3.4 Implementation

The implementation of the POD based deflation can be done in 2 parts. The first part will construct the deflation subspace matrix Z based on the POD basis. The first part requires $m \in \mathbb{N}$ solutions of the previous systems. The solutions are adjusted to match the dimensions of the updated system as described in chapter 8.2. The second part will solve the updated system with the DPCG method. This process can be repeated, if one considers to update the system again. The POD basis is computed by removing the oldest solution and adding the most recent solution. All solutions are adjusted to match the dimensions of the updated system. Removing and adding solutions will ensure that the deflation-subspace matrix contains the same amount of columns and that POD basis uses the most recent information which are relevant. This method is also known as a moving window approach.

Algorithm 7: Assembly of deflation matrix based on POD basis.

```
1 Initialise with a set of snapshots $X = \begin{bmatrix} x_1 & x_2 & \dots x_m \end{bmatrix}$ ;
2 Construct the matrix $R = \frac{1}{m-1} X X^T$ ;
3 Obtain the SVD of the matrix $R = \frac{1}{m-1} U \Sigma \Sigma^T U^T$ ;
4 Construct the deflation matrix with the eigenvectors from $U$ corresponding to
    the $p$ largest eigenvalues $\Phi = \begin{bmatrix} u_1 & u_2 & \dots & u_p \end{bmatrix}$ ;
```

Alternatively if $m << n$, then the following algorithm can be applied:

Algorithm 8: Alternative assembly of deflation matrix based on POD basis.

```
1 Initialise with a set of snapshots $X = \begin{bmatrix} x_1 & x_2 & \dots x_m \end{bmatrix}$ ;
2 Construct the matrix $R = \frac{1}{m-1} X^T X$ ;
3 Obtain the SVD of the matrix $R = \frac{1}{m-1} V \Sigma \Sigma^T V^T$ ;
4 Compute the eigenvectors $U = X V \Sigma^{-1}$ ;
5 Construct the deflation matrix with the eigenvectors from $U$ corresponding to
    the $p$ largest eigenvalues $\Phi = \begin{bmatrix} u_1 & u_2 & \dots & u_p \end{bmatrix}$ ;
```

Let $p \geq m$. The moving window approach algorithm including the updated POD basis with p vectors using at most m snapshots:

Algorithm 9: Moving window approach

```
1  if there are no snapshots then
2      for $t$ = 1, 2 do
3          Solve $A^t x^t = b^t$ with the DPCG method described in algorithm 4;
4          Store $x^t$ ;
5  else
6      Construct the deflation subspace matrix $\Phi$ based on the $m$ recent snapshots
         or all snapshots if less than $\min(m,p)$ with algorithm 7 or 8. If there are less
         than $p$ snapshots, then the same amount of eigenvectors will be used as the
         number of snapshots ;
7      Store $\Phi = \begin{bmatrix} u_1^t & u_2^t & \dots & u_p^t \end{bmatrix}$ ;
8      for $t$ = $m$+1, ..., stopping criteria do
9          Solve $A^t x^t = b^t$ with the DPCG method described in algorithm 4 ;
10         Update the deflation subspace matrix $\Phi$ with the $m$ recent snapshots with
             algorithm 7 or 8 ;
11         Overwrite $\Phi$ with $\Phi = \begin{bmatrix} u_1^t & u_2^t & \dots & u_p^t \end{bmatrix}$ ;
```

If there are at least 2 snapshots and $2 < m$, then all snapshots will be used to construct a deflation matrix. The case for 1 snapshot is trivial.

8.4 Results of the POD deflation method

In this section the POD deflation method will be studied for the ssSNPBLUP model when an update occurs. This can be simulated by reducing the original system. The solutions of the reduced linear systems are obtained with the DPCG method, where the subdomains have size $l = 10$ and are chosen with random sampling as described in chapter 7.4.5. The solutions are used to create a POD basis which in turn is used as a deflation subspace matrix for the deflation method as described in algorithm 8. The original system will be solved with the DPCG method with the deflation vectors obtained from the POD basis.

8.4.1 POD 1% case

In figure 8.13 a POD method has been applied with the solutions of the reduced linear systems with reduction $\{0.25\%, 0.5\%, 0.75\%, 1\%\}$. These solutions are obtained with the DPCG method using random sampling with subdomain size $l = 10$. This results in a deflation subspace matrix with 4 columns. The initial solution taken is the zero vector. The POD deflation method yields a reduction of 32.8% compared with the PCG method using initial solution zero.
The DPCG method using random sampling with subdomain size $l = 10$ where the initial solution is the solution from the previous genetic evaluation (solution from the reduced system with a reduction of 0.25%), yields a reduction of 85.9% iterations compared with the PCG method using initial solution zero. Excluding the computation time, it outper-

forms the POD method.

Comparing the POD method with the PCG method where the initial solution is the solution from the previous genetic evaluation. The reduction in iterations equals 3.4%, which is abysmal. Although, the POD method starts with the zero vector, similar results are obtained when the initial solution is the solution from the previous genetic evaluation. The results with different initial solutions for the POD method are showin in figure 8.15. The DPCG method yields a reduction of 79.3%. Concluding, that it is not advisable to perform the POD deflation method up to 1% over the PCG method.

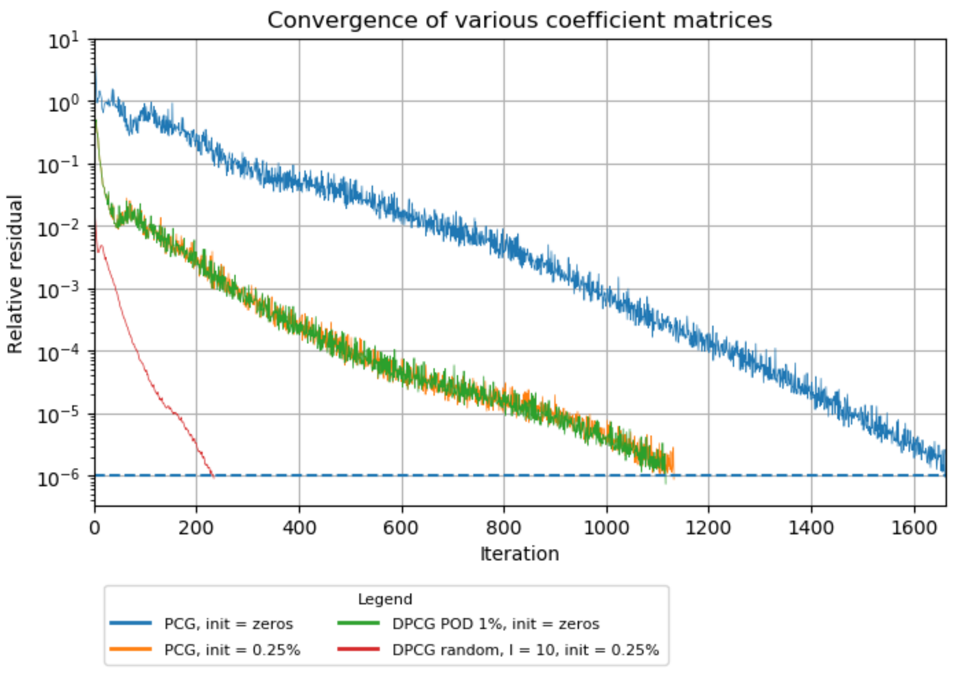


Figure 8.13: Comparison between POD, PCG and DPCG with random sampling using subdomain size $l = 10$.

8.4.2 POD with information up to 3%

Using more information for the POD method could yield an improvement in the convergence rate. POD bases are created based on the solutions of the reduced linear system up to 1%, 2% and 3% with steps of 0.25%. The solutions of these reduced systems are obtained in a similar fashion as the 1% case described earlier. For the POD deflation method, the initial solution is equal to the zero solution.

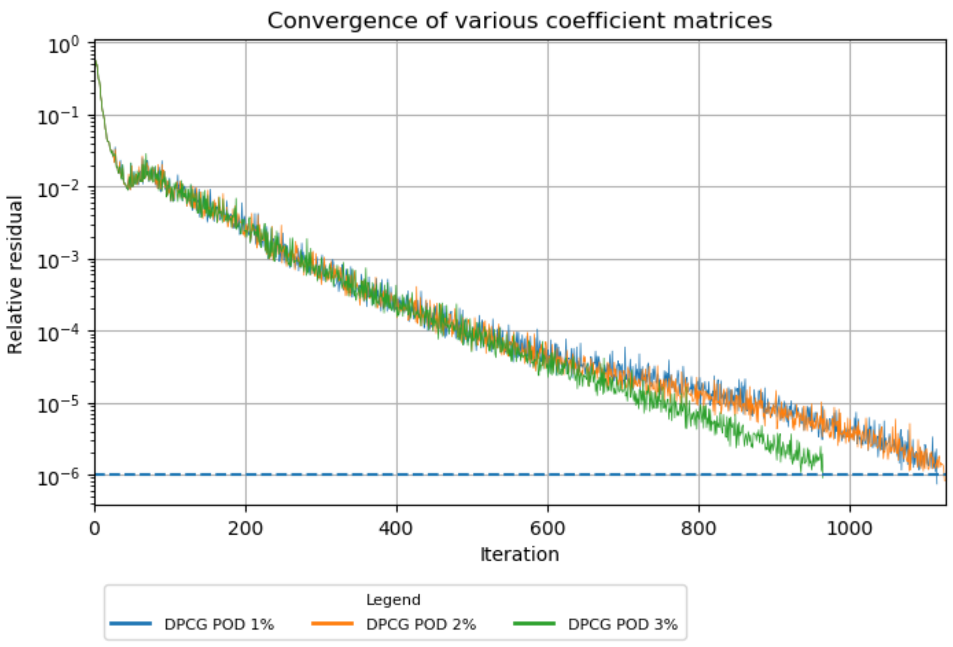


Figure 8.14: Comparison between POD basis with reduced systems up to 1%, 2% and 3%.

It is quite peculiar that there is no improvement for 2%, while the condition number is slightly smaller than 1%. For 3%, there is a relative large improvement for the rate of convergence, as seen in figure 8.14. In table 8.2, it can be observed that the largest eigenvalues seem to remain the same for all POD cases. However, the smallest eigenvalues of the coefficient matrix gets larger when more information is used. The smallest eigenvalue for 3% is twice as large compared with the other POD cases. Therefore, it has the smallest condition number. This result corresponds with the improvement in the rate of convergence. The DPCG POD 3% case compared with the PCG taking the initial solution 0.25% case reduces the amount of iterations by 15%.

Table 8.2: The smallest and largest Ritz values of the deflated preconditioned matrix with a POD basis.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG	0.001435	323.05	226199.1	1662
PCG, init = 0.25%	0.001452	323.05	222449.2	1133
DPCG POD 1%	0.001587	323.04	203608.6	1095
DPCG POD 2%	0.001619	322.97	199432.8	1099
DPCG POD 3%	0.003046	322.93	106002.8	963
DPCG l = 10, init = 0.25%	0.001533	7.11	4636.2	235

8.4.3 POD method sensitivity of the initial solution

Recall from chapter 7.4, that the coefficient matrix (6.26) contains many large eigenvalues due to the SNP effects. Note that the POD method only uses a few deflation vectors and these vectors seem to only affect the smallest eigenvalues as seen in table 8.2. This means that the error of an initial solution containing components of the eigenvectors related to the large eigenvalues can cause slower convergence. This problem should not occur with the DPCG method with subdomain size $l = 10$, which consists of 1236 subdomains. This method mainly affects the large eigenvalues. Additionally, the amount of deflation vectors used for this deflation method, should project relatively many large eigenvalues to 0. Therefore, this deflation method should allow the same convergence rate for a wider range of initial solution compared with the POD method.

To test the sensitivity of the initial solution for the POD method. The POD 1% case will be studied with different initial solutions. The initial solutions are:

- An initial solution with all zeros as entries.
- An initial solution with the solution from the reduced 0.25% system.
- An initial solution with all ones as entries. Motivation of this choice is choosing a solution that does not resemble the solution and does not belong to $\mathcal{R}(Z)$. Also the values corresponding to the SNP effects are relatively large compared with the solution. Thus the error of this initial solution contains components corresponding to the large eigenvalues.

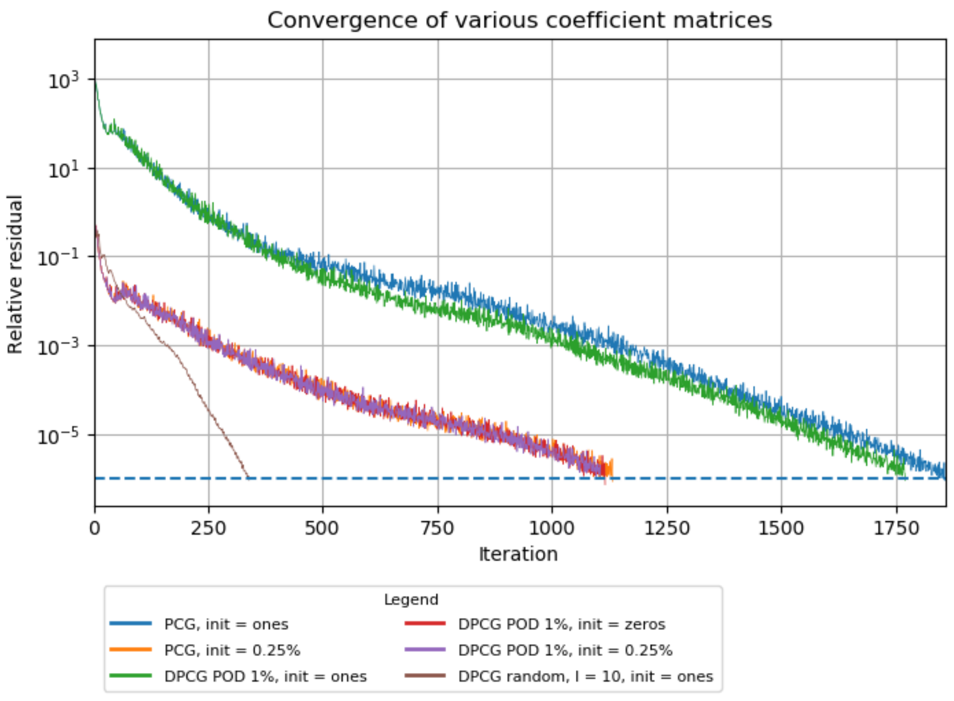


Figure 8.15: Comparison of the POD method with different initial solutions.

From figure 8.15, it can be observed that the POD method with initial solution with all zeros, the POD method with the solution from the previous genetic evaluation and the PCG with method with the solution from the previous genetic evaluation have a similar convergence pattern.

The solution with all ones has a pattern similar to the PCG method also starting with all ones. Recall that the error of this initial solution contains components of eigenvectors corresponding to the large eigenvalues. Thus this result can be expected, since the POD method only affects small eigenvalues.
The performance of the POD method is sensitive to the choice of initial solution for the ssSNPBLUP model.

8.4.4 Comparison between covariance matrix

The POD method can also be applied on the covariance matrix or correlation matrix based on the matrix X. Apply the same definition of X as in chapter 8.3.2, then define the mean of the data set as:

$$\bar{X} = \frac{1}{m}\sum_{i=1}^{m} x_i \ \in \mathbb{R}^n$$

Let $\mathbf{e} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix} \in \mathbb{R}^{1\times n}$ be a matrix with all ones. The mean centred data matrix is defined as:

$$Y = X - \bar{X}\mathbf{e}$$

Then the covariance matrix is defined as:

$$V = \frac{1}{m-1} YY^T$$

and the correlation matrix is defined as

$$W = \frac{1}{m-1}\left(\operatorname{diag}(V)\right)^{-\frac{1}{2}} \cdot V \cdot \left(\operatorname{diag}(V)\right)^{-\frac{1}{2}}$$

Only the results of the covariance matrix is shown here. The results of the correlation matrix are near identical and can been seen in Appendix 10.6. The same data is used as in chapter 8.4.2.

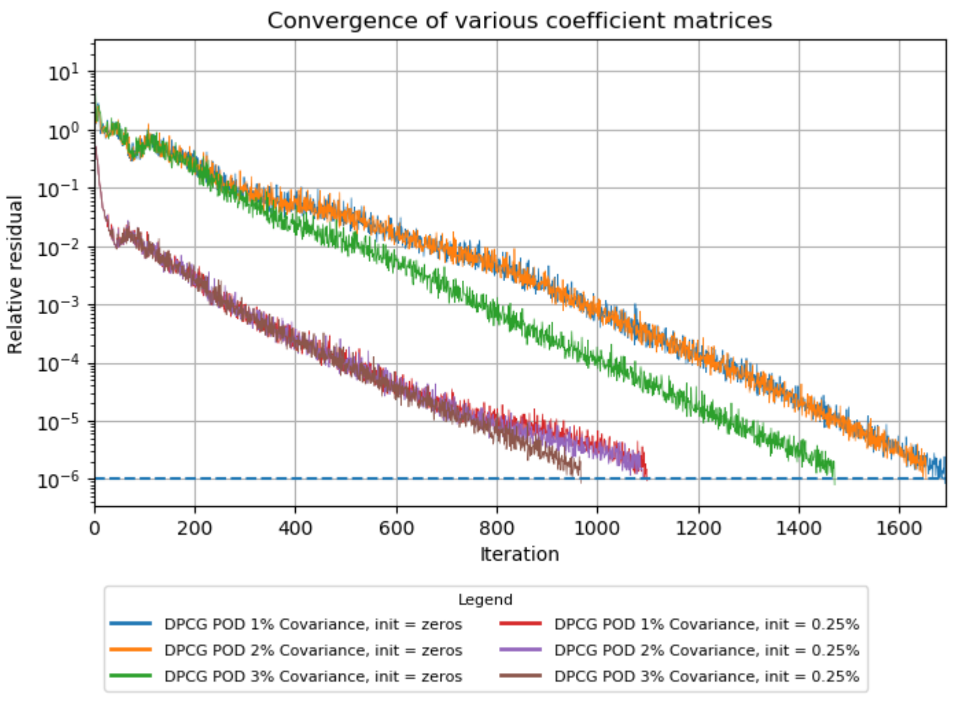


Figure 8.16: Comparison between different POD bases and two different initial solutions.

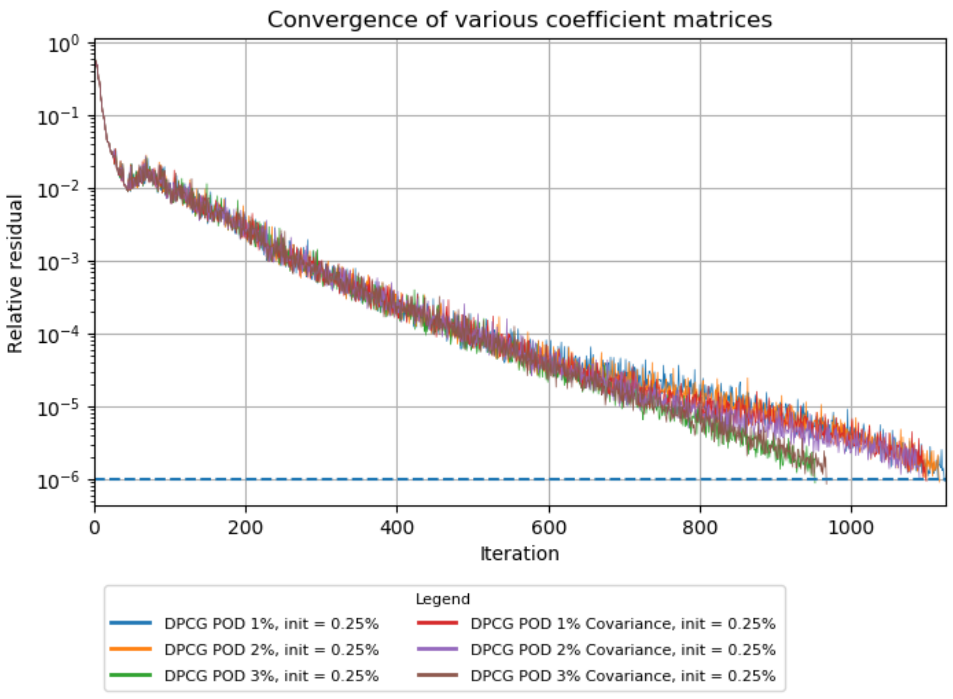


Figure 8.17: Comparison between the data matrix and covariance matrix using the solution from the reduced original system with a reduction of 0.25% as the initial solution.

The convergence behaviour using the covariance matrix is the same as the data matrix when the initial solution is the solution of the previous genetic evaluation, see figure 8.17.

Surprisingly, the convergence behaviour is different when the solution is the zero solution. This behaviour does not occur for the data matrix, compare with figure 8.14. The POD method applied on the covariance matrix or correlation also affects the small eigenvalues in a similar way, compare table 8.2 and 8.3.

Table 8.3: The smallest and largest Ritz values of the deflated preconditioned matrix with a POD basis based on the covariance matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
DPCG POD 1%	0.001563	323.02	206650.4	1100
DPCG POD 2%	0.001742	322.86	185320.8	1091
DPCG POD 3%	0.003019	322.92	106945.8	968

8.4.5 Removing deflation vectors

In table 8.2 and 8.3 a large jump of the smallest Ritz value can be observed. To gain more insight on the behaviour of the Ritz values, additional computations have been made with different POD bases including solutions from the reduced linear system with reductions ranging from 1% to 3% with steps of 0.25%. The results can be seen in figures 8.18-8.20.

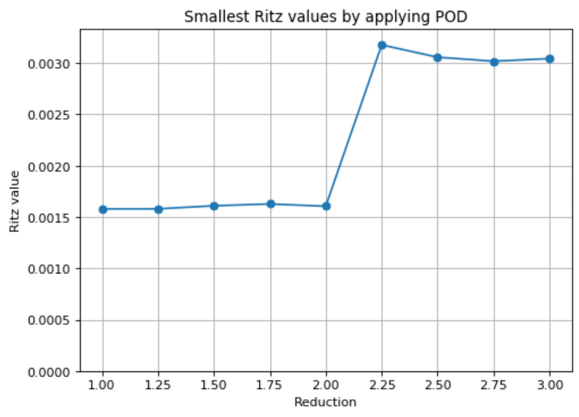


Figure 8.18: Smallest Ritz values of each preconditioned coefficient matrix applying a POD basis deflation method.

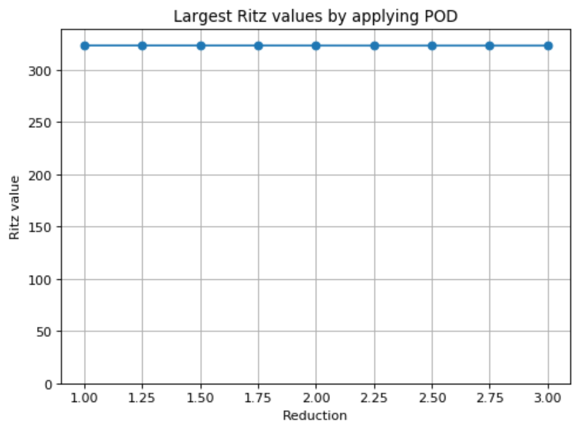


Figure 8.19: Largest Ritz values of each preconditioned coefficient matrix POD basis deflation method.

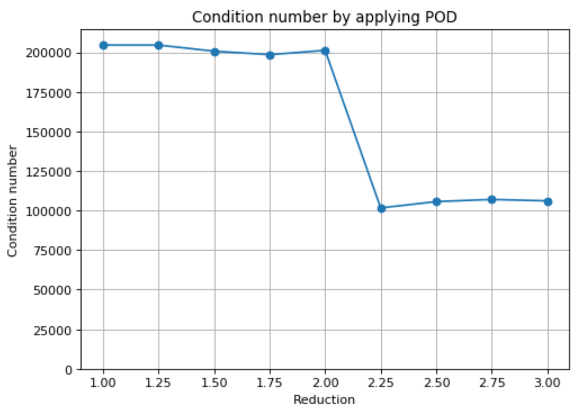


Figure 8.20: Condition number of each preconditioned coefficient matrix applying POD basis deflation method.

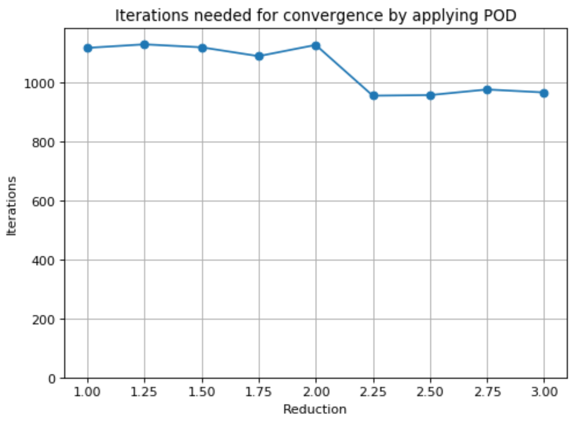


Figure 8.21: Iterations required for convergence.

A jump occurs between 2% and 2.25% reduction. Moreover, the amount of iterations required for convergence is not monotonically decreasing, which is quite unusual. Since using more deflation vectors should yield the same or a better rate of convergence. Although, the amount of iterations needed is not fluctuating significantly between $1\% - 2\%$ and $2.25\% - 3\%$. Thus it could be said that the improvement is stagnating between $1\% - 2\%$ and $2.25\% - 3\%$. Hence adding more deflation vectors, does not necessarily improve the convergence. Similar results have been obtained with the covariance and correlation matrix. This result raises a new question:

Does removing deflation vectors from the POD method yields a similar rate of convergence?

How does one know which vectors to remove, such that an improvement in the convergence is still beneficial? Recall, that the POD method affects the smallest eigenvalue in the ssSNPBLUP model. Moreover, the POD deflation method is applied on the preconditioned coefficient matrix. It could be that a linear combination of the POD vectors approximates the eigenvector(s) corresponding to the smallest eigenvalue(s) of our preconditioned system. By taking the POD vectors that have the largest contribution an approximation can be made with a few vectors instead of all POD vectors. The POD vectors are eigenvectors of the data, covariance or correlation matrix and the eigenvalues can be seen as a measure of variance of the data set in the direction of the corresponding eigenvector. Therefore, selecting POD vectors with the largest eigenvalues is a way to select them based on contribution.
Define the eigenvalues of a matrix as $\lambda_1 \geq \lambda_2, \ldots \geq \lambda_m \geq 0$ with $m \in \mathbb{N}$. The following criterion is applied to determine the amount of deflation vectors needed:

$$\frac{\sum_{i=1}^{p} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq \alpha \tag{8.6}$$

where $\alpha \in \mathbb{R}$ is the threshold and $p \in \mathbb{N}$ is the smallest number such that the criterion holds. The eigenvalues ratio can be defined as:

$$\frac{\lambda_i}{\sum_{i=1}^{m} \lambda_i} \tag{8.7}$$

The eigenvalue ratios of the data, covariance and correlation matrix are shown in figure 8.22.

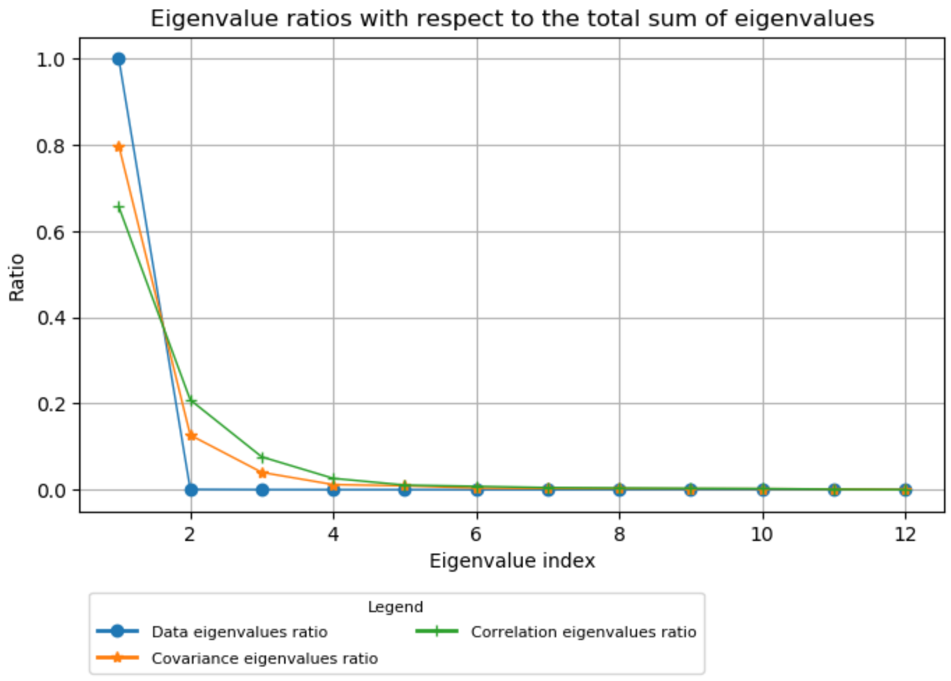


Figure 8.22: Eigenvalue ratios of the data, covariance and correlation matrix defined in equation (8.7).

Applying the criterion shown in equation (8.6) on the data, covariance and correlation matrix with

$$\alpha = 0.99$$

yields the following numbers of eigenvectors used:

Table 8.4: Number of vectors based on the criterion per matrix.

Matrix	Number of vectors
Data	1
Covariance	7
Correlation	8

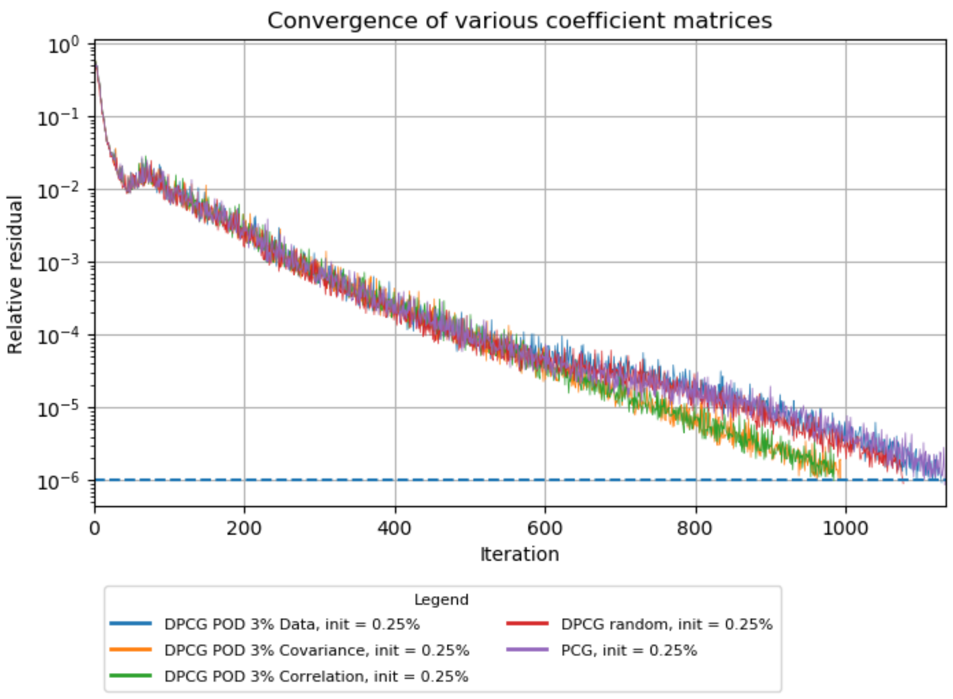


Figure 8.23: Comparison between the method POD method applied on the data, covariance and correlation method including the criterion.

For the following results the same data is used as in the POD 3% case. The POD vector with the largest eigenvalue is picked first as a deflation vector. If the threshold α in equation (8.6) is not satisfied, then the POD vector with the second largest eigenvalue is picked, and so on.

Table 8.5: Ritz values of the POD method applied on the data, covariance and correlation matrix.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
DPCG POD Data	0.001528	323.05	211417.3	1129
DPCG POD Covariance	0.002979	322.96	108427.8	995
DPCG POD Correlation	0.002971	322.95	108705.4	986

The convergence of the POD method applied on the data matrix is similar as using no POD deflation. It can be concluded that the deflation vector used is not a good approximation of the eigenvector corresponding to the smallest eigenvalue.
The POD method applied on the covariance and correlation matrix yields the quickest convergence. The convergence is similar as the POD 3% case seen in figure 8.17. The POD method applied on the covariance matrix utilises 1 deflation vector less than the POD method applied on the correlation matrix. This is quite odd, since the matrices are similar. Also the contribution of this extra deflation vector in the correlation matrix is quite low, see figure 8.22. Moreover, both methods affect the smallest eigenvalue in a similar way as using all POD vectors, this can be observed in table 8.3 and 8.5.

It could be that the criterion introduced in equation (8.6) is not appropriate for this problem. This becomes more evident with the following results were 2 deflation vectors are picked corresponding to the largest eigenvalues. The same data as in chapter 8.4.2 is used. Two deflation vectors are chosen, since the ratios after the second deflation vector are significantly smaller than the first two ratios, see figure 8.22. These matrices are constructed with similar data as the data matrix. Thus similar behaviour could be expected for the data matrix.

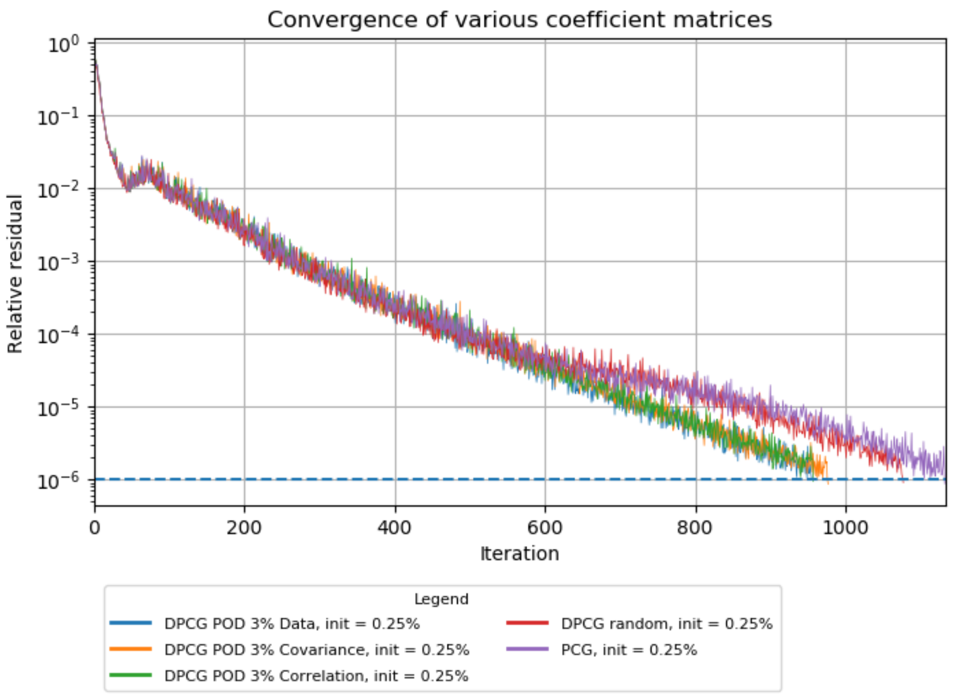


Figure 8.24: POD method applied on the data, covariance and correlation matrix with 2 deflation vectors ($p = 2$).

Table 8.6: Ritz values of the POD method applied on the data, covariance and correlation matrix ($p = 2$).

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
DPCG POD Data	0.003233	323.99	99897.7	956
DPCG POD Covariance	0.003178	322.98	101625.7	978
DPCG POD Correlation	0.003216	322.98	100438.0	958

The POD method applied on the data matrix now converges similarly as the other matrices. The convergence pattern is similar in all cases. The same convergence pattern can also be observed in figure 8.23. Comparing tables 8.5 and 8.6, there are less iterations needed for the data matrix when utilising an additional deflation vector. For the covariance and correlation matrix a slight decrease in iterations for convergence can be observed, by comparing it with the 3% case incorporating the criterion and the case using all deflation vectors. Those results can be seen from tables 8.3 and 8.5. This is quite peculiar, since more deflation vectors are used in those cases. However, the additional deflation vectors have a relatively small eigenvalue ratio, see figure 8.22. Therefore, deflation vectors with a small eigenvalue ratio do not seem to affect the convergence of the DPCG method.

It can be concluded that criterion (8.6) is not appropriate for this problem.

8.4.6 Removing data

In chapter 8.4.5, the results show that not all deflation vectors are required from the POD based deflation method. Thus it could be that some of the data is redundant. The eigenvalues of the data, covariance and correlation matrix can be seen in the Appendix at table 10.13. For all matrices, it can be observed that the first two eigenvalues are significantly larger than the remaining eigenvalues. Therefore, it could be that some of the solutions used for the data is similar or the solution can be a linear combination of the other solutions. Thus if one removes some of the solutions from the data, it could be possible to retain an improvement in convergence.

Define X as the matrix containing solutions at each column. To verify if the data contains linearly independent vectors, the rank of X^TX should be be calculated. This can be done by considering the eigenvalues. These can be obtained from the singular values of the SVD method.
If an eigenvalue is zero, then the data contains one linearly dependent vector. If multiple eigenvalues are zero, then there are several linearly dependent vectors. Due to the machine precision, the following threshold has been set. If an eigenvalue with an absolute value smaller than 10^{-13}, then those are considered to be equal to 0.
Consider the mean centred data defined in chapter 8.4.4. By definition there is at at least 1 linear dependent vector. This can be seen by summing all data vectors together, which results in the zero vector. Thus the mean centred data compared with the unaltered data always has at least 1 additional linear dependent vector.
The eigenvalues of X^TX and $\bar{X}^T\bar{X}$ can be seen in Appendix 10.6. These values are obtained during the POD based deflation method, because those matrices are used for this method. The matrix X^TX has no eigenvalue equal to 0. Thus the solutions are linearly independent. The matrix $\bar{X}^T\bar{X}$ has one singular value equal to 0. This is expected, since all the solutions in the data are mean centred. Hence the unaltered data has no dependent vectors and the mean centred data has 1 dependent vector.

There is at most 1 dependent vector, it does not explain the behaviour shown in figure 8.21, where a relative large change can be observed in iterations at 2.25%. Hence, it could be that the data added from the 2.25% solution contains something essential for an improvement. It could also be the accumulated effect of all data.

To test this, the POD deflation method will use data from the reduced original system with reductions $\{0.25\%, 2.25\%\}$. A correlation plot of these solutions is shown in figure 8.25 to showcase the difference.

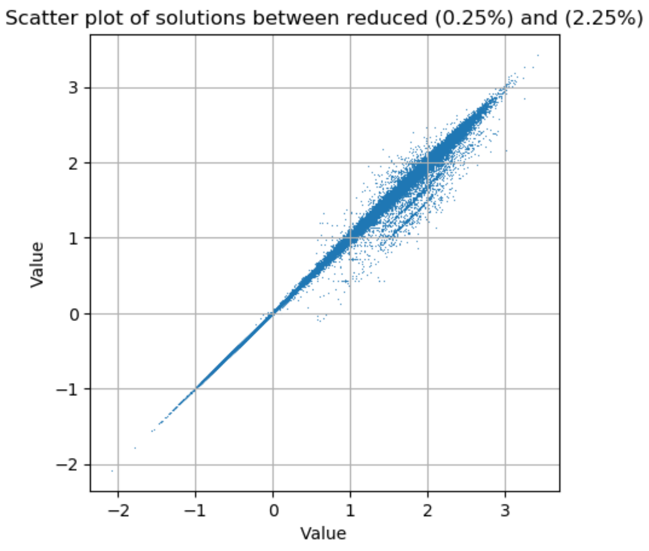


Figure 8.25: Scatter plot with the solution of 0.25% on the x-axis and the solution of 2.25% on the y-axis.

For the following results the initial solution will be the solution of the reduced original system with a reduction of 0.25%. The POD based deflation method applied on the reduced data set where all deflation vectors are used, yields almost no improvement, see figure 8.26.

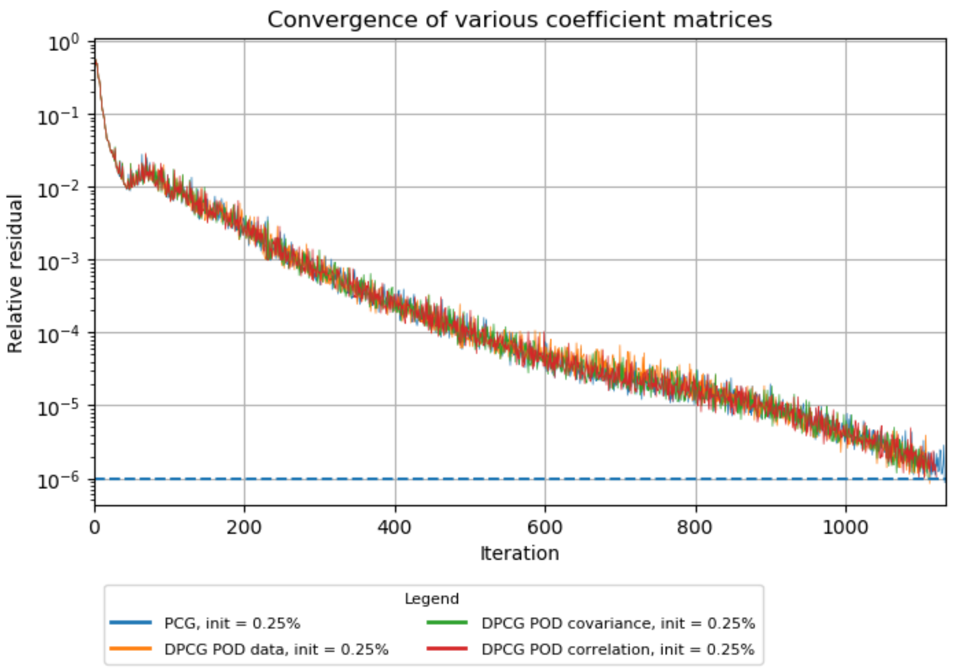


Figure 8.26: Convergence of the POD based deflation method with 2 deflation vectors.

Thus the solution corresponding to the reduction of 2.25% does not seem to contain essential data for a significant improvement.

Surprisingly, an improvement in convergence can be seen with the data set including solutions corresponding with reductions $\{0.25\%, 3\%\}$, see figure 8.28. Compared with the PCG method with initial solution corresponding to the solution of the reduced 0.25% system, a reduction in iterations of 11.7% for the data matrix, 13.4% for the covariance matrix and 14.4% for the correlation matrix can be observed.

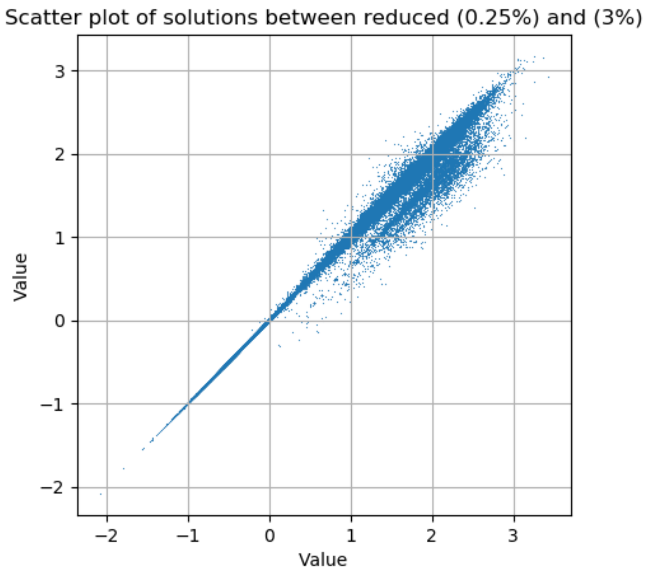


Figure 8.27: Scatter plot with the solution of 0.25% on the x-axis and the solution of 3% on the y-axis.

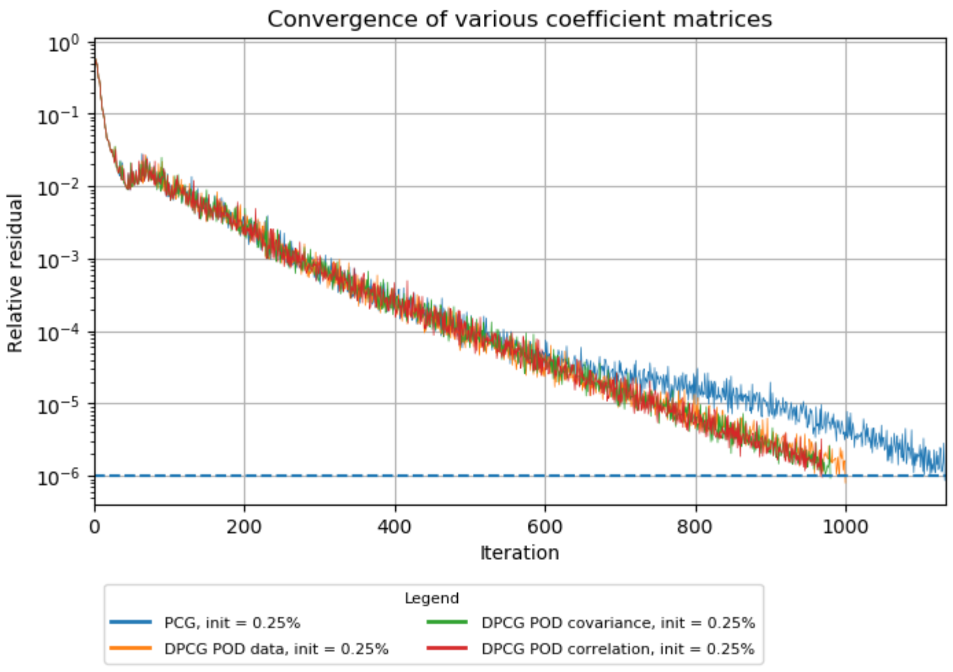


Figure 8.28: Convergence of the POD based deflation method with 2 deflation vectors.

From an earlier explanation at the start of this section 8.4.6, it is mentioned that the mean centred data set contains at least 1 dependent vector. Thus one mean centred data vector can be removed. This vector can be any of the data vectors. Thus for the POD deflation method applied on the covariance or correlation matrix 1 less data vector is required.
In the previous case of data containing solutions of $\{0.25\%, 3\%\}$ and after mean centring, the mean centred solution corresponding to 3% has been removed. This could have also been the 0.25% solution. Therefore, the POD method applied on the covariance and correlation matrix only considers 1 vector.
Consider the data vectors x_1 and x_2. Define the matrix $X = \begin{bmatrix} x_1 \end{bmatrix}$ and $\bar{X} = \frac{x_1+x_2}{2}$. The deflation vector resulting from the covariance matrix can be expressed algebraically from the SVD decomposition of

$$X - \bar{X} = U\Sigma V^T$$

Consider the matrix $\left(X - \bar{X}\right)^T \left(X - \bar{X}\right)$. The SVD decomposition of this matrix can be defined as:

$$\begin{aligned} \left(X - \bar{X}\right)^T \left(X - \bar{X}\right) &= V\Sigma\Sigma^T V^T \\ &= 1 \cdot \left[\left(X - \bar{X}\right)^T \left(X - \bar{X}\right)\right]^2 \cdot 1 \end{aligned}$$

In the last equation, it is important to note that the matrix on the left-hand side is a scalar. Additionally, the eigenvector has dimension 1 and should have norm equal to 1. Thus the eigenvector equals 1. The eigenvector of the covariance matrix equals.

$$\begin{aligned} U &= \left(X - \bar{X}\right) V\Sigma^{-1} \\ &= \frac{1}{\left(X - \bar{X}\right)^T \left(X - \bar{X}\right)} \left(X - \bar{X}\right) \end{aligned}$$

Thus the deflation vector is the mean centred data vector scaled by its own squared norm. Analogously, the deflation vector corresponding to the correlation matrix can be derived.

In the case of two data vectors, this results in choosing the deflation vector with the largest eigenvalue resulting from the POD basis applied on the covariance and correlation matrix, since the other eigenvalue equals 0. Remarkably, for the data matrix the deflation vector with the largest eigenvalue does not improve the convergence, but the deflation vector corresponding to the second largest eigenvalue does improve the convergence. Results of this case are not shown in this thesis, but a comparable case of the data matrix can be seen in figure 8.23.

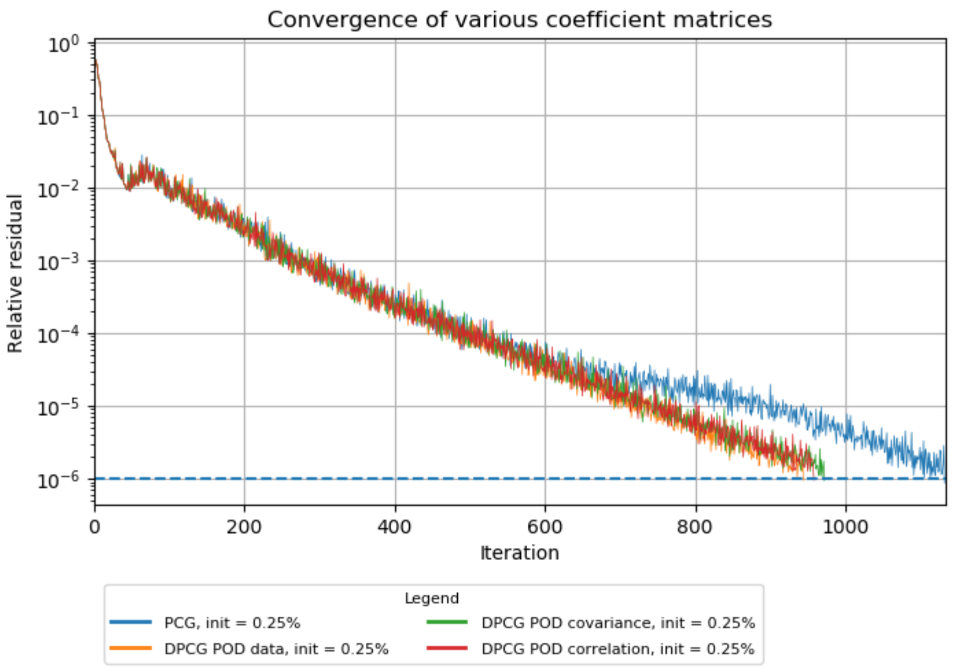


Figure 8.29: Convergence of the DPCG POD based deflation method with 1 deflation vector.

Comparing figures 8.28 and 8.29, it can be observed that applying 1 deflation vector needs slightly less iterations for convergence. The amount of iterations and effect on the smallest Ritz value are similar for the case $p = 2$ described in chapter 8.4.5. The results can be seen in table 8.6 and 8.7.

Table 8.7: Ritz values of the POD based deflation method from data including $\{0.25\%, 2.25\%\}$ and $\{0.25\%, 3\%\}$.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG init, 0.25%	0.001452	323.05	222449.2	1133
Data $\{0.25\%, 2.25\%\}$	0.001556	322.98	207521.9	1113
Covariance $\{0.25\%, 2.25\%\}$	0.001489	322.95	216859.9	1111
Correlation $\{0.25\%, 2.25\%\}$	0.001486	322.95	217373.5	1122
Data $\{0.25\%, 3\%\}$	0.002721	322.99	118685.2	1001
Covariance $\{0.25\%, 3\%\}$	0.003131	322.91	103122.0	981
Correlation $\{0.25\%, 3\%\}$	0.003155	322.91	102346.5	970
Data $\{0.25\%, 3\%\}$ and $p = 1$	0.003400	322.99	94985.9	945
Covariance $\{0.25\%, 3\%\}$ and $p = 1$	0.003158	322.99	102285.8	973
Correlation $\{0.25\%, 3\%\}$ and $p = 1$	0.003186	322.99	101371.3	959

The deflation vector resulting from the data matrix and covariance matrix are similar to each other, which can be seen in figure 8.30. The deflation vector resulting from the correlation matrix should be similar as well, since it can be derived in a similar fashion as the covariance matrix. Therefore, the result of the correlation matrix has not been reported in thesis.

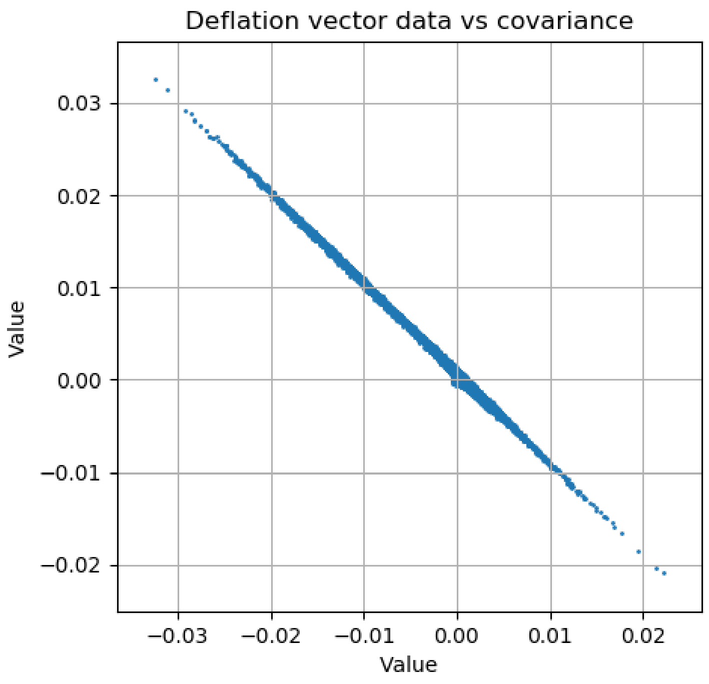


Figure 8.30: Comparsion between thge deflation vectors resulting from the data and covariance matrix.

8.4.7 Comparison with eigenvector

The POD method applied on data that is based on the solutions of the reduced original system seems to affect the smallest eigenvalue. A possible conclusion would be that the span of the chosen deflation vectors contains the eigenvector corresponding to the smallest eigenvalue. In chapter 8.4.6, the amount of deflation vectors required for an improvement in convergence has been reduced to 1 vector. Thus comparing this deflation vector with the eigenvector of interest from the preconditioned system, should give an answer if the span of the deflation vectors contains the eigenvector of interest.

To approximate the eigenvector corresponding to the smallest eigenvalue, the Lanczos algorithm has been used, see chapter 8 (p.132, [26]) for more details on the algorithm. The first 25 smallest eigenvalues of the preconditioned system are shown in figure 8.31.

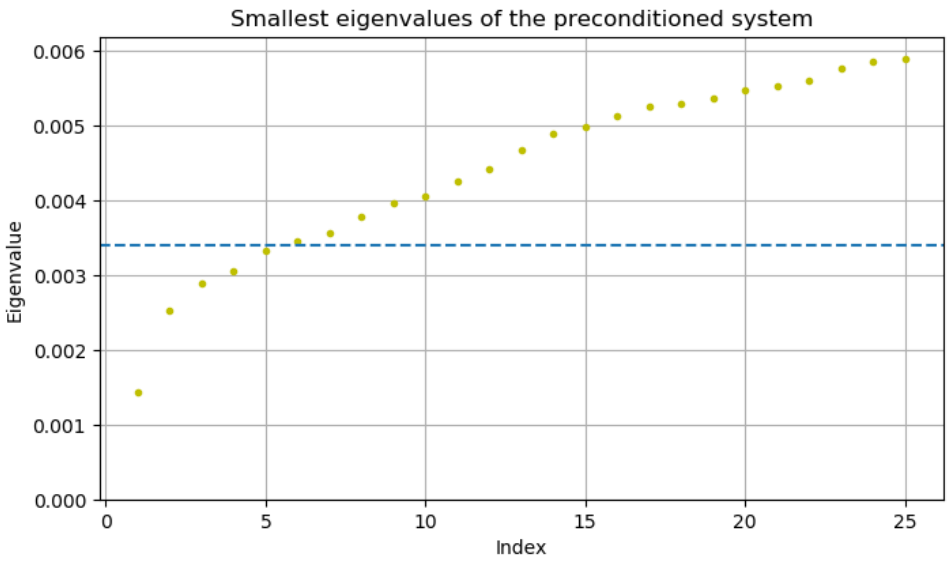


Figure 8.31: The first 25 smallest eigenvalues of the preconditioned system. The horizontal line takes on the value 0.0034, which is the largest minimum Ritz value observed in table 8.7.

From table 8.7 it can be seen that the smallest Ritz value in the case with 1 deflation vector has a value around $0.0031 - 0.0034$. Thus the deflation vector and the combination with the chosen initial solution which is the solution of the reduced (0.25%) system affects the first 4 or 5 smallest eigenvalues.

To verify whether the deflation vector of the data, covariance and correlation matrix is a linear combination of the eigenvectors corresponding to the first 25 smallest eigenvalues, the same method as in chapter 8.4.6 to verify linear dependent vector is used. Thus a matrix containing the eigenvectors and the deflation vector of interest is constructed. This matrix has full rank for the deflation vector of the data, covariance and correlation matrix. Therefore, the deflation vectors for all 3 matrices are not a linear combination of the eigenvectors corresponding to the first 25 smallest eigenvalues.
Henceforth, the effect on the smallest eigenvalues is a combination of the deflation method and the chosen initial solution. The error of this initial solution probably contains no components of the eigenvectors coinciding with the small eigenvalues.

8.4.8 POD method combined with subdomain decomposition

In general, the deflation method is effective if the eigenvectors corresponding to the extremal eigenvalues are contained in the span of the deflation vectors. The POD deflation vectors affects only the smallest eigenvalues and the subdomain decomposition methods shown in chapter 7.4 affects mostly the largest eigenvalues of the preconditioned system. A potential improvement in covergence can be obtained, if the deflation vectors of the POD method are not contained in the span of the deflation vectors of the subdomain decomposition method.

The deflation vectors used in chapter 7.4.5 with subdomain size $l = \{10, 100\}$ and the deflation vectors from the data matrix used in chapter 8.4.2 are combined. Two different initial solutions have been applied, which are the zero solution and the solution of the reduced original system with a reduction of 0.25%.

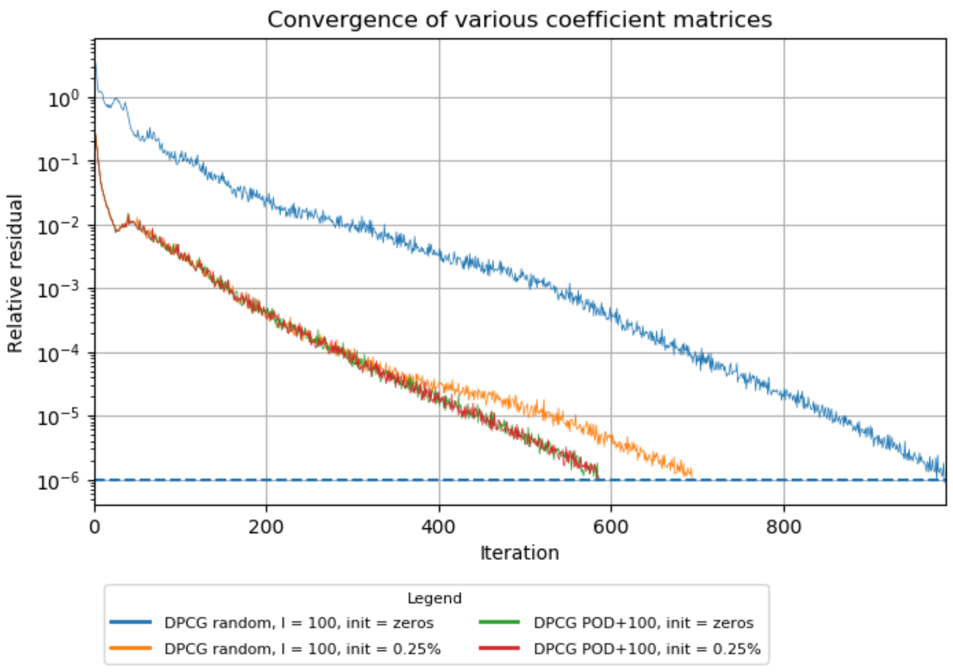


Figure 8.32: Comparison between POD and DPCG with random sampling using subdomain size $l = 100$, which is denoted as POD+100 in the legend.

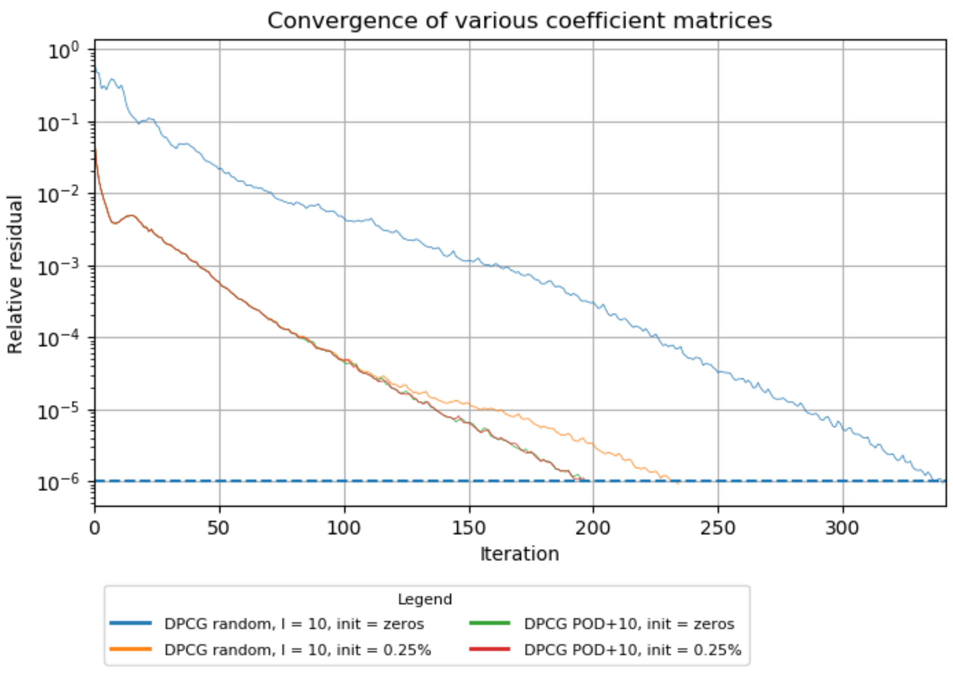


Figure 8.33: Comparison between POD and DPCG with random sampling using subdomain size $l = 10$, which is denoted as POD+10 in the legend.

From figures 8.32 and 8.33, it can be clearly seen that the combination of the POD method and the subdomain decomposition method yields an improvement in the rate of convergence.

Table 8.8: For the sake of brevity, the **POD+ method** refers to the combination of POD method and Random Sampling method. Ritz values of the POD+ method including all solutions from 0.25% up to 3% with steps of 0.25%. The computation times of the bolded methods can be seen in table 8.9.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
PCG init 0.25%	0.001452	323.05	222449.2	1133
$l = 100$	0.001497	78.02	52102.9	988
$l = 100$, **init** 0.25%	0.001512	78.02	51609.9	695
POD+ $l = 100$	0.003097	77.91	25158.0	586
POD+ $l = 100$ **init** 0.25%	0.003094	77.91	25183.6	587
$l = 10$	0.001500	7.11	4738.3	341
$l = 10$, **init** 0.25%	0.001533	7.11	4636.2	235
POD+ $l = 10$	0.003228	7.10	2200.9	197
POD+ $l = 10$ **init** 0.25%	0.003215	7.10	2209.4	198

The iterations required for the POD + random sampling method are the same for both initial solutions. Furthermore, the Ritz values are the same in both initial solution cases.
For $l = 100$, the POD + random sampling method yields a reduction of 15.5% compared with just only applying the subdomain decomposition method using an initial solution

of 0.25%. For $l = 10$, a reduction of 15.7% can be observed.
Thus applying the random sampling method for subdomain decomposition and including the POD vectors does improve the rate of convergence in iterations. However, the overhead of the POD method during the assembly of the deflation subspace matrix should also be considered. The computation times of the bolded methods seen in table 8.8 are shown in table 8.9. The inclusion of the POD method results in a quicker iteration process, since less iterations are required and this is due to the condition number being roughly twice as small. Additionally, this reduction in iteration time outweighs the overhead of the assembly of the deflation subspace matrix. From this it should be clear, that the POD method is a beneficial addition for the deflation method.

Table 8.9: Computation times of the DPCG method by random sampling the SNPs without the chromosome restriction and including deflation vectors from the POD method. Time is shown in **seconds**. The column with the name **Total** is the sum of **Deflation Matrix** and **Iteration**. The row in bold has the quickest total computation time.

Method	Deflation Vectors	Deflation Matrix	Iteration	Total
PCG init 0.25%	0	0	1884.13	1884.13
$l = 100$, init 0.25%	125	1.76	1208.06	1209.82
POD+ $l = 100$ init 0.25%	137	4.53	1022.85	1027.38
$l = 10$, init 0.25%	1236	1.80	479.74	481.54
POD+ $l = 10$ init 0.25%	**1248**	**4.61**	**430.02**	**434.63**

8.4.9 POD and k-means++ clustering

In practice, multiple genetic evaluations are done with the same SNP genotype profile of animals. Thus the k-means algorithm discussed in chapter 7.4.6 will be a favourable choice to combine it with the POD method. The POD method will only create 2 deflation vectors. This method is explained in chapter 8.4.5. Moreover, the POD method will be applied on the data matrix. Results of the covariance matrix can been seen in Appendix 10.6. In this section, the same labelling of names to denote the number of clusters for the k-means algorithm as in chapter 7.4.6 is applied, see table 7.6. The results from the combination of the POD and k-means++ algorithm are shown in the following figures and tables.

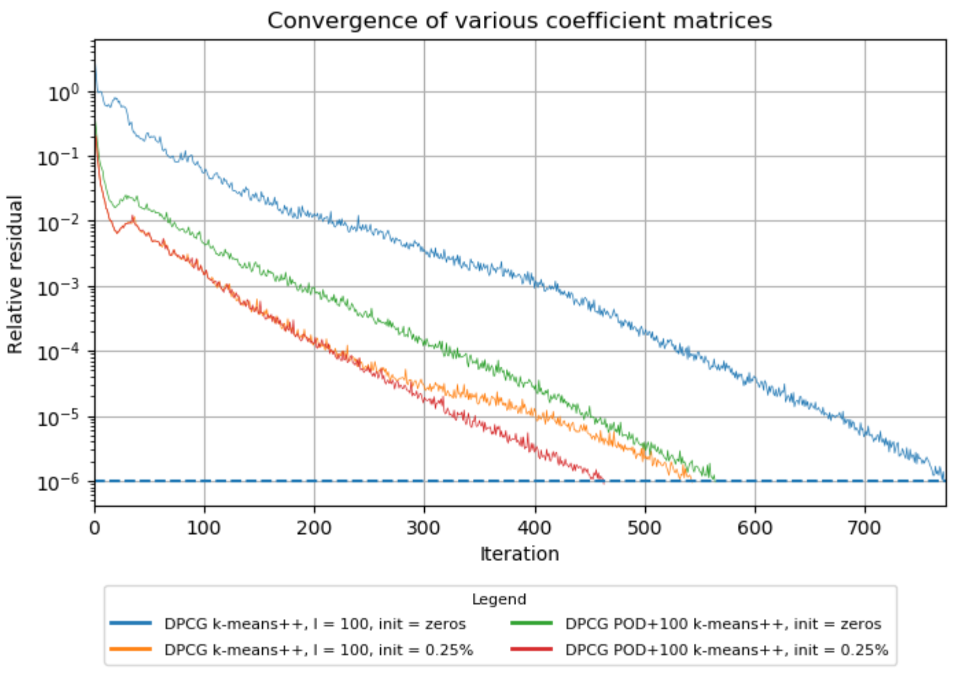


Figure 8.34: Comparison between POD + k-means++ using subdomain size $l = 100$.

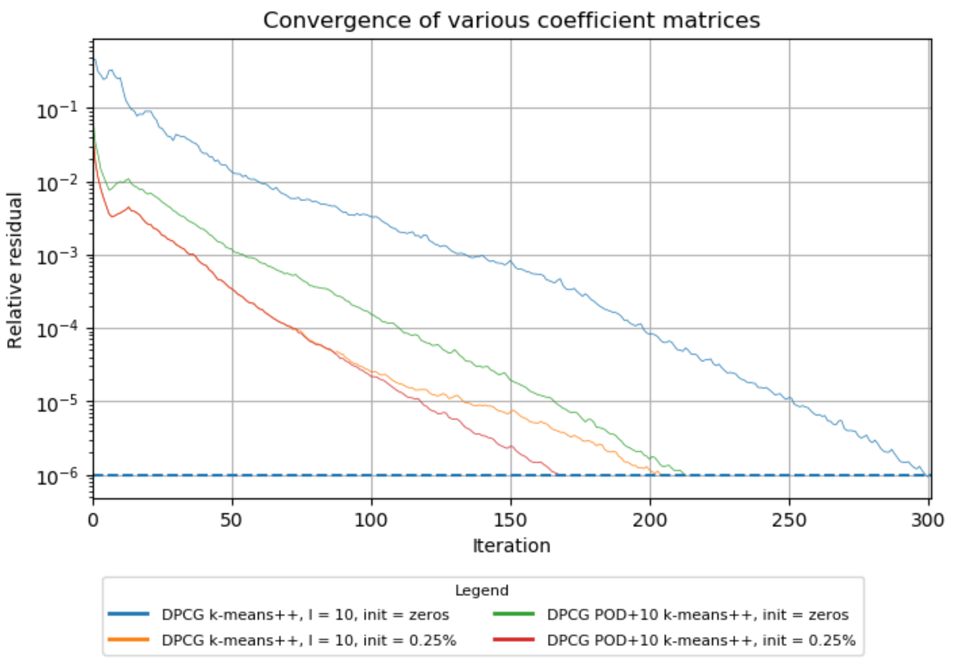


Figure 8.35: Comparison between POD + k-means++ using subdomain size $l = 10$.

Table 8.10: Ritz values of the POD + k-means method including only all solutions from 0.25% and 3%.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
k-means++ $l = 100$	0.001498	48.41	32321.6	775
k-means++ $l = 100$, init 0.25%	0.001515	48.41	31949.7	544
POD+ k-means++ $l = 100$	0.002584	48.39	18727.3	565
POD+ k-means++ $l = 100$ init 0.25%	0.003306	48.39	14636.5	464
k-means++ $l = 10$	0.001503	5.74	3818.0	303
k-means++ $l = 10$, init 0.25%	0.001546	5.74	3712.45	205
POD+k-means++ $l = 10$	0.002614	5.74	2194.7	214
POD+k-means++ $l = 10$ init 0.25%	0.003626	5.74	1582.5	169

The results are similar as the cases shown in table 8.8. In general the k-means algorithm needs less iterations for convergence with a similar amount of deflation vectors compared with the case in chapter 8.4.8, because the condition number of the deflated preconditioned coefficient matrix is smaller.
The inclusion of the POD method with 2 deflation vectors seems to be only working well for the initial solution 0.25%. This could be due to the difference in the condition number. For the initial solution equal to the zero solution, the difference lies in the smallest Ritz value, which is smaller than the other initial solution case. This is quite odd, because the same deflation subspace matrix has been applied. This means that the same extremal eigenvalues are expected. Note that the Ritz values are obtained from the DPCG method through the Lanczos method. Thus it is quite likely that the smallest Ritz value did not converge yet in the initial solution 0.25% case. Henceforth, it is possible that the error of the initial solution 0.25% could have no components of some eigenvectors corresponding to the smallest eigenvalues.

Table 8.11: Computation times of the DPCG method by the k-means++ algorithm and including 2 deflation vectors from the POD method. Time is shown in **seconds**. The row in bold has the quickest total computation time.

Method	**Deflation Vectors**	**Deflation Matrix**	**Iteration**	**Total**
$l = 100$, init 0.25%	124	146.55	1173.55	1320.10
POD+ $l = 100$ init 0.25%	126	172.46	777.78	950.24
$l = 10$, init 0.25%	1235	561.39	417.59	978.98
POD+ $l = 10$ init 0.25%	**1237**	**570.01**	**349.26**	**919.27**

If one excludes the overhead time of the assembly of the deflation-subspace matrix. Then by comparing the iteration columns in tables 8.9 and 8.11, the POD method with 2 deflation vectors including the k-means++ algorithm $l = 10$ seems to achieve the fastest computation time. Overall, the POD method combined with the k-means++ approach performs better at the iteration process if the initial solution corresponds to the solution of the reduced system with a reduction of 0.25%.

8.5 Concluding remarks

In chapter 8.2 the initial solution is a rough approximation of the original system. This can be constructed by approximating the estimated breeding value (EBV) of animals that are missing in the previous genetic evaluation, then these approximations are included in the solution of the previous genetic evaluation. By using this solution as the initial solution a reduction in iterations of approximately 30% has been observed.

The POD deflation method has been proposed in chapter 8.3.2. This method creates a few deflation vectors from a specific set of data. This data consists of the solutions from the previous genetic evaluations. These previous genetic evaluations are simulated by removing the most recent of phenotype and genotype records of animals from the original data set, this process will be referred as reduction. The reductions are done with steps of 0.25% up to a total reduction of 3%. The EBV of the missing animals are approximated with the same method described in chapter 8.2.
The POD method seems to be quite sensitive of the used data. Solutions of the reduced systems up to 2% did not seem to improve the convergence. However, after adding results from 2.25% a sudden improvement could be noticed, see chapter 8.4.5. Moreover, the choice of initial solution can greatly affect the convergence of the POD deflation method, especially in the case of the covariance and correlation matrix when all data is used including all deflation vectors resulting from the POD method. This sensitivity of the initial solution also occurs for the data matrix, when not all data and deflation vectors are used. The best results are observed when the initial solution of 0.25% is used, this chosen initial solution resembles the solution from the most recent genetic evaluation.
If the solution from the 0.25% reduced system is used as the initial solution, then not all deflation vectors are required to obtain the same improvement in the rate of convergence. A criterion based on the ratio of the eigenvalues of the data, covariance and correlation matrix (8.6) between the total sum of the eigenvalues has been applied to determine the amount of deflation vectors. However, this criterion does not seem to be robust for all matrices used for the POD method, since the data matrix led to no improvement in convergence with this criterion. Also results in chapters 8.4.5 and 8.4.6 show that only 1 deflation vector is required for a similar improvement in the rate of convergence.
If the range of the deflation subspace matrix contains eigenvectors corresponding to the problematic eigenvalues, then an improvement in convergence can be expected. Although, it turns out that the deflation vector is not a linear combination of the eigenvectors corresponding to the 25 smallest eigenvalues of the preconditioned system. Combining this fact with the sensitivity of the initial solution, it could be that the error of the initial solution contains no components of eigenvectors corresponding to the smallest eigenvalues.

The POD deflation method affects the smallest eigenvalues of the preconditioned coefficient matrix, whilst the subdomain decomposition method in chapter 7.4 only affects the largest eigenvalues. Combining the deflation vectors of the POD and subdomain decomposition method yields a deflation matrix that affect both sides of the eigenvalue spectrum. The effectiveness of this combination is similar to using the POD method and subdomain decomposition separately. Thus it can be concluded that the deflation vectors resulting from the POD deflation method are not linear combinations of the subdomain

decomposition method.
The k-means++ combined with the POD approach has the fastest iteration process with the same amount of deflation vectors as the random sampling with the POD approach. This is due to the fact that the condition number is smaller by applying the k-means++ algorithm with the same amount of deflation vectors as for the random sampling method. Furthermore, this effect on the condition number is also consistent with the results shown in 7.4.6. Moreover, a reduction of 81% in iteration time can be observed from the best performing combination of the POD and k-means++ (1235 deflation vectors) approach by comparing it with the PCG method where both methods use the previous genetic evaluation as an initial solution. Whilst considering the computation time of the whole process of the aforementioned methods (assembly of the coefficient matrix and deflation matrix, and the iteration process) a reduction of roughly 49% in time can be observed. A similar comparison can be done with the combination of the POD and k-means++ approach, and the POD and random sampling approach. The k-means++ yields a reduction in iteration time of roughly 19% and an increase of 112% in total computation time. The increase in time is due to the overhead of the k-means clustering algorithm. The overhead of the assembly of the deflation subspace matrix from the k-means algorithm is significantly larger than the random sampling method. In general, the random sampling approach yielded the fastest total time. However, for multiple genetic evaluations, the deflation vectors resulting from the k-means clustering algorithm have to be only computed once, because the same SNP genotype profile is used. By comparing the computation times of the two different subdomain decomposition strategies combined with the POD method, a calculation can be made that shows how many evaluations are required to outweigh the cost of the overhead of the deflation subspace matrix assembly process. For these calculations the initial solution corresponds to the solution of the 0.25% reduced system, the random sampling method has a fixed subdomain size of $l = 10$ and the k-means algorithm has a comparable amount of deflation vectors which equals the number 1235.

$$\begin{aligned}&\left|\frac{\text{Deflation assembly time Random Sampling} - \text{Deflation assembly time k-means++}}{\text{Iteration time Random Sampling - Iteration time } k\text{-means++}}\right| \\ &= \frac{|4.61 - 570.01|}{430.02 - 349.26} \\ &\approx 7\end{aligned}$$

Thus 7 additional genetic evaluations have to be done to outweigh the cost of the produced overhead from the k-means++ algorithm. In this very specific case, if 7 or more additional genetic evaluations are required, than the k-means++ algorithm is favourable.

CHAPTER 9

Conclusion

A two-level preconditioner applied on the ssSNPBLUP model has been investigated. The simulated data that is used for this research resulted in a coefficient matrix that is SPD. Recall the main research question:

Why does the deflation method applied on the preconditioned system of the ssSNPBLUP model improves the convergence?

From the research done in this thesis a concrete answer to this question is not possible yet. However, in the case of the subdomain decomposition deflation method a motivation can be given. In chapter 7.3 a relationship between the ssSNPBLUP and ssGBLUP model can be seen, where the ssSNPBLUP is equivalent to the ssGBLUP model if the subdomain size equals 1. For this case the least amount of iterations are required for convergence, although the size of the Galerkin matrix makes this choice of subdomain size impractical, because this results in more work for the DPCG method than the PCG method. Moreover, if the subdomain size increases, then it will take more iterations for convergence. This trend is seen for all subdomain decomposition strategies discussed in chapter 7. **Therefore if the subdomain size decreases, it seems that subdomain decomposition deflation method converges to the rate of convergence as the ssGBLUP model.**
It is important to note that the subdomain decomposition deflation method only affects the largest eigenvalues.

To reduce the size of the Galerkin matrix less deflation vectors are needed. Hence, the subdomain size should be increased. Consequently, this means that the rate of convergence will decrease compared with the subdomain size equal to 1 case. Therefore, a different method to choose the subdomains has to be applied, which will answer the following research sub question:

Does it matter how subdomains are constructed for the convergence of the DPCG method applied on the ssSNPBLUP model?

In this thesis the k-means algorithm has been proposed in chapter 7.4.6. This method clusters the SNPs based on the correlations among them. The results in the aforementioned chapter shows that the k-means algorithm needs **roughly half** the deflation

vectors to converge with the same rate as the random sampling method with and without restricting the SNPs on a chromosome, and following the map of the chromosomes method, these methods are studied in chapter 7. **Hence it does matter how subdomains are chosen.**
Also 3 variations of the k-means algorithm have been studied. The first approach uses random initial centres. The second approach makes use of the SNP arrangement of the data by choosing the SNPs equidistantly based on indices. The last approach is the k-means++ algorithm. In all cases, a similar convergence behaviour is shown. Thus there is no obvious choice in which approach is the best. The k-means algorithm could converge to a local minimum, because of poorly chosen initial centres, while the k-means++ algorithm tries to choose the initial centres better by spreading out the initial centres, which means the k-means++ algorithm will be more robust in theory.

In the case of updating the ssSNPBLUP model with new data, the following research question has to be answered:

How can you accelerate the convergence of the updated system and why does it work?

The updates are simulated by reducing the original system. The reduced systems should resemble previous genetic evaluations. A reduction of up to 3% with steps of 0.25% has been applied by removing the most recent phenotype and genotype records from the original data. This step size could differ from what happens in practice, but it is usually smaller than the step size used here.
Two main things have been studied. The first one is choosing an initial solution and the second one is a POD based deflation method that utilises information of the previous systems.
In the case of the initial solution, a simple method has been applied, see chapter 7.4.3. This method chooses the initial solution as the solution from the previous genetic evaluation. This solution misses values of animals (EBV) that were not present in the old data. Thus the missing values (EBV) are approximated by taking the average of the EBV of the parents. **It turns out that the initial solution closely resembles the solution of the updated the system**. Thus the initial solution is "close" to the real solution. Hence an improvement on the rate of convergence can be expected. A reduction of 30% in iterations can be achieved for this particular data set.
In chapter 8.3.2 the POD based deflation method has been investigated. The POD method analyses a matrix based on a specific data set for our model, which are the solutions from the previous systems. The solutions of the previous systems do not contain EBVs of the new animals after an update. These EBVs are approximated in the same way as in the study of the initial solution in chapter 7.4.3. Three different matrices based on the data that includes solutions from the previous evaluations have been studied, which are the data, covariance and correlation matrix. Deflation vectors are obtained by taking the eigenvectors corresponding to a non-zero eigenvalue of one of the three matrices. The data matrix yielded slightly different results than the covariance and correlation matrix. This happens when the initial solution is different than the solution of the previous genetic evaluation. Comparing this case with the initial solution as the solution from the previous genetic evaluation, the data matrix seems to not be affected by this choice, but the covariance and correlation matrices do. Thus the data matrix seems to be less sensi-

tive to the choice of the initial solution. Therefore, the data matrix is more robust.
Also adding more data does not guarantee an improvement in convergence. This is possibly due to the fact that some of the data is too similar with each other.
This lead to the investigation on removing deflation vectors and removing data vectors. The amount of deflation vectors could be reduced from 12 deflation vectors to 1 deflation vector, which retained the same rate of convergence. Although, the rate of convergence becomes more sensitive on the choice of the initial solution. A reduction of 15% in iterations is reported compared with the PCG method starting with the solution of the previous genetic evaluation. Thus the POD deflation method only requires a few deflation vectors to affect the rate of convergence. Moreover, the POD deflation method affects the smallest eigenvalues. It is not quite clear why the POD method affects the smallest eigenvalues. A hypothesis is that the POD method picks up the large absolute differences at the EBV components between the solutions from the previous genetic evaluations. This is due to the fact that the other components which relates to the SNPs have relatively smaller values. For the subdomain decomposition method it has been observed that only changing the SNP components of the deflation vector affects the largest eigenvalues. Thus the information that the POD method catches, could be related to the eigenvectors corresponding to the smallest eigenvalues. However, it has been reported that the deflation vectors are not a linear combination of the first 25 eigenvectors corresponding to the smallest eigenvalues.
Recall that the subdomain decomposition deflation method affects the largest eigenvalues. Applying the subdomain decomposition and POD method, and combining the deflation vectors results in a deflation method that affects both ends of the eigenvalue spectrum. The results show that a subdomain decomposition method with the addition of the POD method accelerates the convergence, see chapter 8.4.8.

All in all, the method with the **best improvement** in the rate of convergence compared with the PCG method is the combination of **the POD method**, **the k-means subdomain decomposition approach** and **the choice of the initial solution as the previous genetic evaluation**. However, the POD method can only be employed if more than 1 solution from previous genetic evaluations are available. Likewise, the method to choose the initial solution can only be used if the solution of a previous genetic evaluation is known. In a scenario with none of these data available, the k-means approach is the method of choice. However, the subdomain decomposition methods applied on the ssSNPBLUP model have the inherit problem that the method requires many deflation vectors, this could be an issue with the memory usage of the DPCG algorithm. Moreover, the k-means approach has an overhead due to the k-means algorithm, this overhead is negligible if multiple genetic evaluations are done.

Further research

In this research several new methodologies are proposed with a favourable effect on the convergence. However, there are still many questions unanswered. For example, whether the methods proposed are robust or even a better alternative can be used. The next recommendations could lead to answers for these examples.

Different data

It is advisable to test the same methodology on different data with at least one of the following characteristics: *multiple fixed effects* or *multiple traits*. In this thesis only the case with 1 fixed effect and 1 trait has been studied. By applying more than 1 fixed effect, the coefficient matrix resulting from the ssSNPBLUP model becomes an SPSD matrix.

Identifying important clusters and different clustering methods

In this research the k-means clustering algorithm has been applied once over all SNPs. In figures 7.29 and 7.31 a few large sized cluster outliers can be seen. Even if the amount of cluster increases this trend stays, but the large cluster sizes do decrease in size in an absolute sense. Combining this with the fact that the rate of convergence increases when the number of clusters increases, it could be that the large clusters play a lesser role for the improvement on the rate of convergence. Perhaps a coarse clustering can be applied first and then a finer clustering on the important clusters. This poses another interesting issue, which is to identify clusters that are important.
Additionally, different clustering methods on SNPs can be investigated, preferably a method that can utilise the biological properties of SNPs (e.g., haplotypes).
An alternative could be a method that can detect SNPs that are linked with each other. In mathematical terminology that would be finding a set of SNPs that are statistically associated. An example is to adjust the k-means algorithm with a different measure instead of the 2-norm.

Approximations for the initial solution

In the case of an update in the data, the initial solution is chosen as the solution from the previous genetic evaluation. The solution of the previous genetic evaluation does not contain values of the new animals. In this research only one approach to approximate the missing values has been applied, which is by simply taking the average EBV of the parents of the animal. To see whether this approach is a good method, it can be compared with different approaches to approximate the missing values. These approaches are yet to be determined. In [10] a method to estimate SNP effects and EBVs of genotyped animals with no phenotype records are shown. The SNP effect estimation is applied with a different purpose in mind, but it could inspire for the usage in estimating SNP effects for the initial solution.
Moreover, these initial solutions can be seen as an approximation of the solution of the updated system. Therefore, the initial solution can be represented as a perturbed solution of the updated system. Hence, the effect of a perturbation on the convergence of the DPCG method can be studied. This could potentially give a clearer insight on how to approximate the missing values.

Robustness of the methodology with a different stopping criterion

In a recent paper of Vandenplas et al. [23] a new stopping criterion has been proposed. The new stopping criterion can determine more accurately when to stop the PCG and

DPCG method. Applying this criterion results in less iterations. One could question if the methodologies applied in this thesis also holds up with the new stopping criterion.

CHAPTER 10

Appendix

10.1 Derivation PCG

The preconditioned linear system that is of our interests, has the following form

$$\tilde{A}\tilde{x} = \tilde{b} \tag{10.1}$$

where $\tilde{A} = P^{-1}AP^{-T}$, $x = P^{-T}\tilde{x}$, $\tilde{b} = P^{-1}b$ and P is a nonsingular matrix. The preconditoner is the matrix M defined by $M = PP^T$. Moreover, the following relations hold

$$M = M^T \tag{10.2}$$

$$M^{-1} = P^{-T}P^{-1} \tag{10.3}$$

$$r^k = P\tilde{r}^k \tag{10.4}$$

$$p^k = P^{-T}\tilde{p}^k \tag{10.5}$$

The PCG algorithm is obtained by applying the CG algorithm on the linear system (10.1). Hence the algorithm is given by

Algorithm 10: Preconditioned Conjugate Gradient method

```
1 Choose $\tilde{x}^0$; $\tilde{r}^0 = \tilde{b} - \tilde{A}\tilde{x}^0$; $\tilde{p}^0 = \tilde{r}^0$
2 for $k = 0, 1, \dots,$ until convergence do
3     $\tilde{w}^k = \tilde{A}\tilde{p}^k$;
4     $\tilde{\alpha}_k = \frac{<\tilde{r}^k,\tilde{r}^k>}{<\tilde{p}^k,\tilde{w}^k>}$;
5     $\tilde{x}^{k+1} = \tilde{x}^k + \tilde{\alpha}_k\tilde{p}^k$;
6     $\tilde{r}^{k+1} = \tilde{r}^k - \tilde{\alpha}_k\tilde{w}^k$;
7     $\tilde{\beta}_k = \frac{<\tilde{r}^{k+1},\tilde{r}^{k+1}>}{<\tilde{r}^k,\tilde{r}^k>}$;
8     $\tilde{p}^{k+1} = \tilde{r}^{k+1} + \tilde{\beta}_k\tilde{p}^k$;
```

It is possible to rearrange the formulas such that the vectors have no tildes. Let $y^k = M^{-1}r^k$. Note that we can solve $My^k = r^k$ to obtain y^k, instead of applying the inverse of M. Combinining the previous statement about y^k, (10.2), (10.3), (10.4) and (10.5), we have

$$\begin{aligned}
\tilde{\alpha}_k &= \frac{<\tilde{r}_k, \tilde{r}_k>}{<\tilde{p}_k, \tilde{A}\tilde{p}_k>} \\
&= \frac{(r^k)^T P^{-T} P^{-1} r^k}{(\tilde{p}^{k-1})^T P^{-1} A P^{-T} \tilde{p}^k} \\
&= \frac{(r^k)^T M^{-1} r^k}{p^k A p^k} \\
&= \frac{(r^k)^T y^k}{p^k A p^k} \qquad (10.6)
\end{aligned}$$

$$\begin{aligned}
\tilde{\beta}_k &= \frac{<\tilde{r}_{k+1}, \tilde{r}_{k+1}>}{<\tilde{r}_k, \tilde{r}_k>} \\
&= \frac{(r^k)^T P^{-T} P^{-1} r^k}{(r^{k-1})^T P^{-T} P^{-1} r^{k-1}} \\
&= \frac{(r^k)^T M^{-1} r^k}{(r^{k-1})^T M^{-1} r^{k-1}} \\
&= \frac{(r^k)^T y^k}{(r^{k-1})^T y^{k-1}} \qquad (10.7)
\end{aligned}$$

$$\begin{aligned}
\tilde{p}^{k+1} &= \tilde{r}^{k+1} + \tilde{\beta}_k \tilde{p}^k \\
\Longleftrightarrow \quad P^{-T}\tilde{p}^{k+1} &= P^{-T}\tilde{r}^{k+1} + \tilde{\beta}_k P^{-T}\tilde{p}^k \\
\Longleftrightarrow \quad p^{k+1} &= P^{-T}P^{-1}r^{k+1} + \tilde{\beta}_k p^k \\
&= M^{-1} r^{k+1} + \tilde{\beta}_k p^k \\
&= y^{k+1} + \tilde{\beta}_k p^k \qquad (10.8)
\end{aligned}$$

$$\begin{aligned}
\tilde{r}^{k+1} &= \tilde{r}^k - \tilde{\alpha}_k \tilde{A}\tilde{p}^k \\
\Longleftrightarrow \quad P^{-T}\tilde{r}^{k+1} &= P^{-T}\tilde{r}^k - \tilde{\alpha}_k P^{-T}\tilde{A}\tilde{p}^k \\
\Longleftrightarrow \quad P^{-T}P^{-1}r^{k+1} &= P^{-T}P^{-1}r^k - \tilde{\alpha}_k P^{-T}P^{-1}AP^{-T}\tilde{p}^k \\
\Longleftrightarrow \quad M^{-1}r^{k+1} &= M^{-1}r^k - \tilde{\alpha}_k M^{-1}Ap^k \\
\Longleftrightarrow \quad r^{k+1} &= r^k - \tilde{\alpha}_k A p^k \qquad (10.9)
\end{aligned}$$

$$\begin{aligned}
\tilde{x}^{k+1} &= \tilde{x}^k + \tilde{\alpha}_k \tilde{p}^k \\
\Longleftrightarrow \quad P^{-T}\tilde{x}^{k+1} &= P^{-T}\tilde{x}^k + \tilde{\alpha}_k P^{-T}\tilde{p}^k \\
\Longleftrightarrow \quad x^{k+1} &= x^k + \tilde{\alpha}_k p^k \qquad (10.10)
\end{aligned}$$

Combining equations (10.6), (10.7), (10.8), (10.9) and (10.10) yields the following algorithm

Algorithm 11: Preconditioned Conjugate Gradient method

```
1  Choose $x^0$; $r^0 = b - Ax^0$; $p^0 = r^0$;
2  Solve $My^0 = r^0$ and set $p^0 = y^0$;
3  for $k = 0, 1, \ldots,$ until convergence do
4      $w^k = Ap^k$;
5      $\alpha_k = \frac{<r^k, y^k>}{<p^k, w^k>}$;
6      $x^{k+1} = x_k + \alpha_k p^k$;
7      $r^{k+1} = r^k - \alpha_k w^k$;
8      Solve $My^{k+1} = r^{k+1}$ ;
9      $\beta_k = \frac{<r^{k+1}, y^{k+1}>}{<r^k, y^k>}$;
10     $p^{k+1} = y^{k+1} + \beta_k p^k$;
```

10.2 Derivation ssSNPBLUP subdomain decomposition deflation operator

In this section the derivation of the deflation operator applied on the following linear system will be shown:

$$Ax = b \iff \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

The deflation operator is described by the deflation-subspace matrix Z. The deflation-subspace matrix can be constructed by separating the SNP and non-SNP effects into different subdomains. Within the SNP and non-SNP effects an arbitrary grouping can be chosen for smaller subdomains. The deflation-subspace matrix can be defined as:

$$Z = \begin{bmatrix} Z_1 & 0 \\ 0 & Z_2 \end{bmatrix}$$

where Z_1 and Z_2 are of conformable sizes such that the Galerkin matrix is defined as:

$$E = Z^T A Z = \begin{bmatrix} Z_1^T A_{11} Z_1 & Z_1^T A_{12} Z_2 \\ Z_2^T A_{21} Z_1 & Z_2^T A_{22} Z_2 \end{bmatrix}$$

Applying block-inversion on E ([16], see chapter 14.2), the inverse of the Galerkin matrix is given as:

$$E^{-1} = \begin{bmatrix} Q & -Q Z_1^T A_{12} Z_2 C \\ -C Z_2^T A_{21} Z_1 Q & T \end{bmatrix}$$

with

$$\begin{aligned} C &= (Z_2^T A_{22} Z_2)^{-1} \\ Q &= \left(Z_1^T \left(A_{11} - A_{12} Z_2 C Z_2^T A_{21} \right) Z_1 \right)^{-1} \\ T &= C + C Z_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 C \end{aligned}$$

To verify if this is correct, one can work out $E^{-1}E$ or EE^{-1} which should result into the identity matrix. Deriving $E^{-1}E$ results gives the following matrix:

$$E^{-1}E = \begin{bmatrix} \epsilon_{11} & \epsilon_{12} \\ \epsilon_{21} & \epsilon_{22} \end{bmatrix}$$

where

$$\begin{aligned}
\epsilon_{11} &= QZ_1^T A_{11} Z_1 - QZ_1^T A_{12} Z_2 C Z_1^T A_{21} Z_1 \\
&= QZ_1^T \left(A_{11} - A_{12} Z_2 C Z_2^T A_{21}\right) Z_1 \\
&= I \\
\epsilon_{12} &= QZ_1^T A_{12} Z_2 - QZ_1^T A_{12} Z_2 C Z_2^T A_{22} Z_2 \\
&= Q\left(Z_1^T A_{12} Z_2 - Z_1^T A_{12} Z_2 C Z_2^T A_{22} Z_2\right) \\
&= Q\left(Z_1^T A_{12} Z_2 - Z_1^T A_{12} Z_2 I\right) \\
&= 0 \\
\epsilon_{21} &= -CZ_2^T A_{21} Z_1 Q Z_1^T A_{11} Z_1 + TZ_2^T A_{21} Z_1 \\
&= -CZ_2^T A_{21} Z_1 Q Z_1^T A_{11} Z_1 + \left(C + CZ_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 C\right) Z_2^T A_{21} Z_1 \\
&= CZ_2^T A_{21} Z_1 Q Z_1^T \left(A_{12} Z_2 C Z_2^T A_{21} Z_1 - A_{11} Z_1\right) + CZ_2^T A_{21} Z_1 \\
&= CZ_2^T A_{21} Z_1 Q Z_1^T \left(A_{12} Z_2 C Z_2^T A_{21} - A_{11}\right) Z_1 + CZ_2^T A_{21} Z_1 \\
&= -CZ_2^T A_{21} Z_1 I + CZ_2^T A_{21} Z_1 \\
&= 0 \\
\epsilon_{22} &= -CZ_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 + TZ_2^T A_{22} Z_2 \\
&= -CZ_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 + \left(C + CZ_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 C\right) Z_2^T A_{22} Z_2 \\
&= CZ_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 \left(I - CZ_2^T A_{22} Z_2\right) + CZ_2^T A_{22} Z_2 \\
&= CZ_2^T A_{21} Z_1 Q Z_1^T A_{12} Z_2 \left(I - I\right) + I \\
&= I
\end{aligned}$$

Substituting this result yields:

$$E^{-1}E = \begin{bmatrix} I & 0 \\ 0 & I \end{bmatrix}$$

Therefore verifying that E^{-1} is the inverse of E. The deflation matrix is defined as

$$P = I - AZE^{-1}Z^T$$

The deflated coefficient matrix can be derived as follows:

$$\begin{aligned}
PA &= \left(I - AZE^{-1}Z^T\right) A \\
&= A - AZ \begin{bmatrix} Q & -QZ_1^T A_{12} Z_2 C \\ -CZ_2^T A_{21} Z_1 Q & T \end{bmatrix} Z^T A \\
&= A - A \begin{bmatrix} Z_1 Q Z_1^T & -Z_1 Q Z_1^T A_{12} Z_2 C Z_2^T \\ -Z_2 C Z_2^T A_{21} Z_1 Q Z_1^T & Z_2 T Z_2^T \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \\
&= \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} - \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix}
\end{aligned}$$

with

$$\begin{aligned}
\alpha_{11} &= Z_1QZ_1^TA_{11} - Z_1QZ_1^TA_{12}Z_2CZ_2^TA_{21} \\
\alpha_{12} &= Z_1QZ_1^TA_{12} - Z_1QZ_1^TA_{12}Z_2CZ_2^TA_{22} \\
\alpha_{21} &= Z_2TZ_2^TA_{21} - Z_2CZ_2^TA_{21}Z_1QZ_1^TA_{11} \\
\alpha_{22} &= Z_2TZ_2^TA_{22} - Z_2CZ_2^TA_{21}Z_1QZ_1^TA_{12}
\end{aligned}$$

Then substituting this results yields:

$$PA = \begin{bmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{bmatrix}$$

with

$$\begin{aligned}
\rho_{11} &= A_{11} - (A_{11}\alpha_{11} + A_{12}\alpha_{21}) \\
\rho_{12} &= A_{12} - (A_{11}\alpha_{12} + A_{12}\alpha_{22}) \\
\rho_{21} &= A_{21} - (A_{21}\alpha_{11} + A_{22}\alpha_{21}) \\
\rho_{22} &= A_{22} - (A_{21}\alpha_{12} + A_{22}\alpha_{22})
\end{aligned}$$

Rearranging the terms yields:

$$\begin{aligned}
\rho_{11} =& A_{11} - A_{12}Z_2CZ_2^TA_{21} \\
&- \left(A_{11} - A_{12}Z_2CZ_2^TA_{21}\right) Z_1 \left(Z_1^TQZ_1\right)^{-1} Z_1^T \left(A_{11} - A_{12}Z_2CZ_2^TA_{21}\right) \\
\rho_{12} =& A_{12}\left(I - Z_2CZ_2^2A_{22}\right) - A_{11}Z_1QZ_1^TA_{12}\left(I - Z_2CZ_2^TA_{22}\right) \\
&+ A_{12}Z_2CZ_2^TA_{21}Z_1QZ_1^TA_{12}\left(I - Z_2CZ_2^TA_{22}\right) \\
\rho_{21} =& \left(I - A_{22}Z_2CZ_2^T\right)A_{21} - \left(I - A_{22}Z_2CZ_2^T\right)A_{21}Z_1QZ_1^TA_{11} \\
&+ \left(I - A_{22}Z_2CZ_2^T\right)A_{21}ZQZ_1^TA_{12}Z_2CZ_2^TA_{21} \\
\rho_{22} =& A_{22}\left(I - Z_2CZ_2^TA_{22}\right) - A_{21}Z_1QZ_1^TA_{12}\left(I - Z_2CZ_2^TA_{22}\right) \\
&+ A_{22}Z_2CZ_2^TA_{21}Z_1QZ_1^TA_{12}\left(I - Z_2CZ_2^TA_{22}\right)
\end{aligned}$$

If $Z_2 = I$ and conformable with A_{22}, then $C = A_{22}^{-1}$. Substituting it into the deflated coefficient matrix yields:

$$\begin{bmatrix} S - SZ_1\left(Z_1^TSZ_1\right)^{-1}Z_1^TS & 0 \\ 0 & 0 \end{bmatrix}$$

with

$$S = A_{11} - A_{12}A_{22}^{-1}A_{21}$$

10.3 Supplementary k-means algorithm results

The results of the k-means algorithm that have been left out in the main part of the report can be found here.

k-means algorithm

Results of the following number of clusters can be found here $\{5, 61, 1234, 4115\}$ by applying the k-means algorithm using random initial cluster centres.

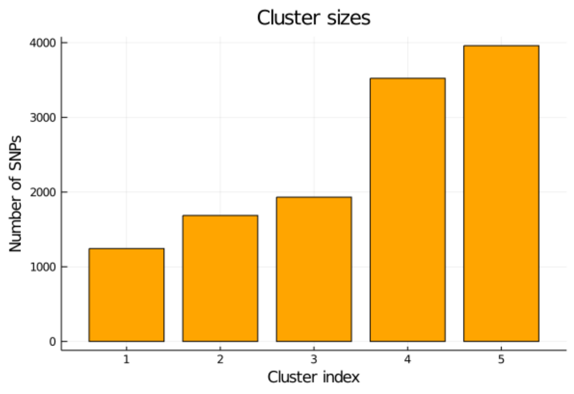


Figure 10.1: All cluster sizes with 5 cluster.

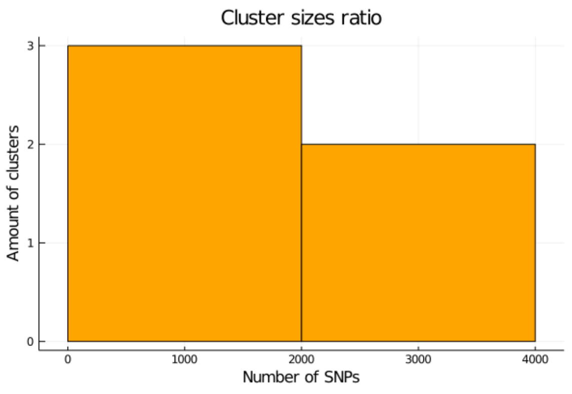


Figure 10.2: Histogram of the cluster sizes.

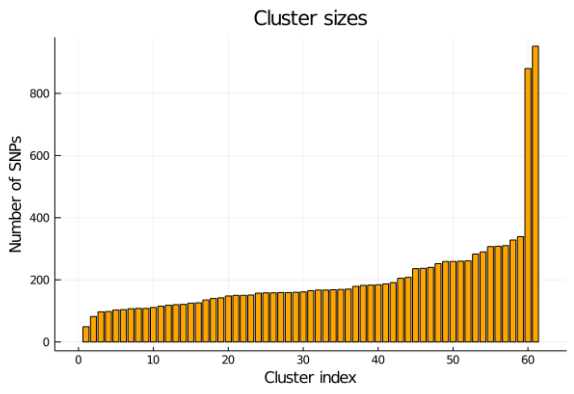


Figure 10.3: All cluster sizes with 61 clusters.

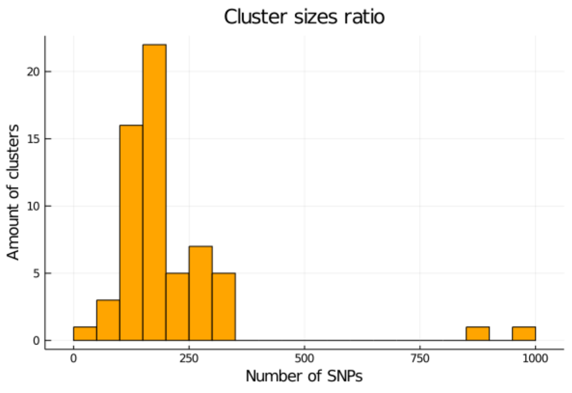


Figure 10.4: Histogram of the cluster sizes.

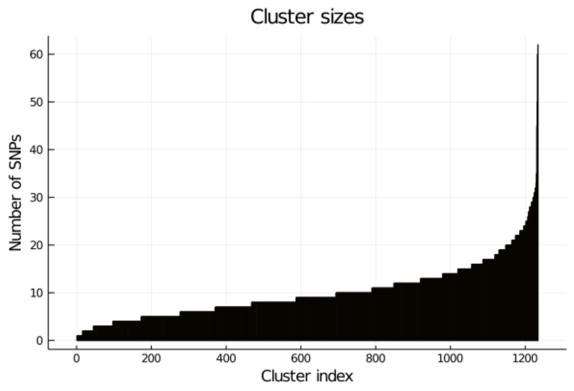


Figure 10.5: All cluster sizes with 1234 clusters.

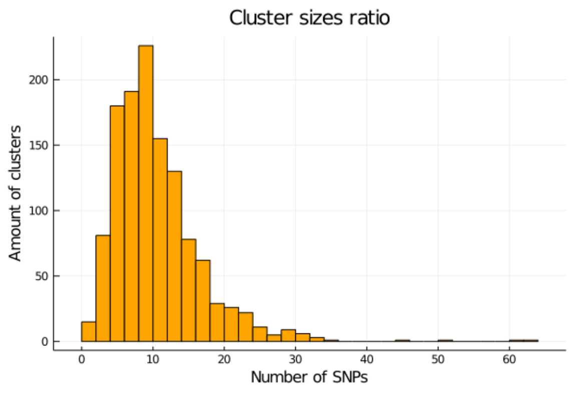


Figure 10.6: Histogram of the cluster sizes.

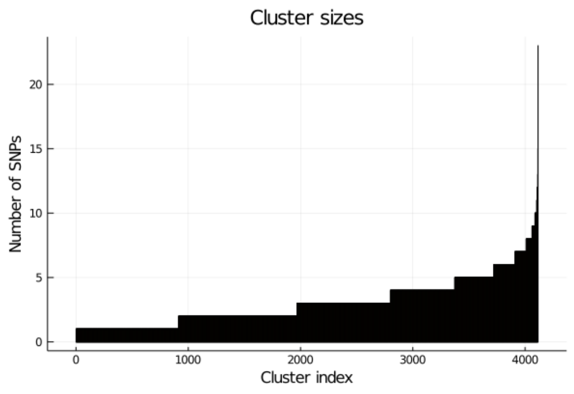


Figure 10.7: All cluster sizes with 4115 clusters.

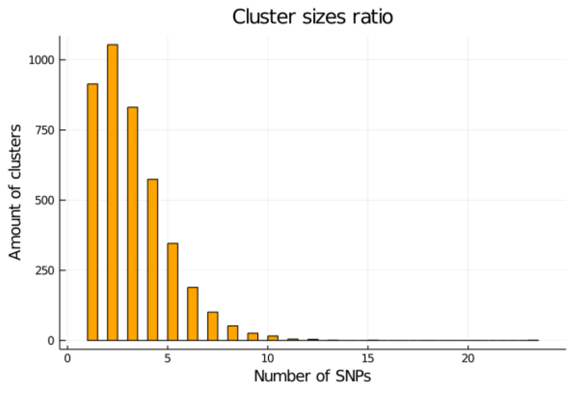


Figure 10.8: Histogram of the cluster sizes.

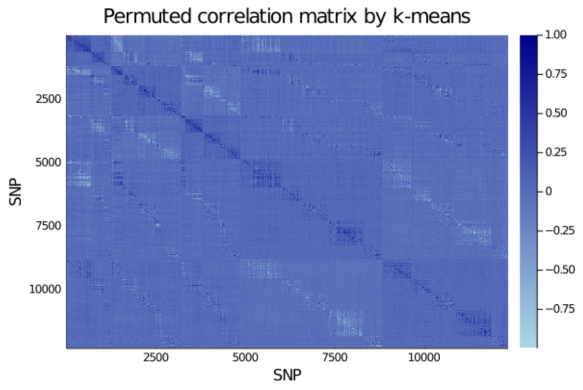


Figure 10.9: Correlation of the SNPs with 5 clusters.

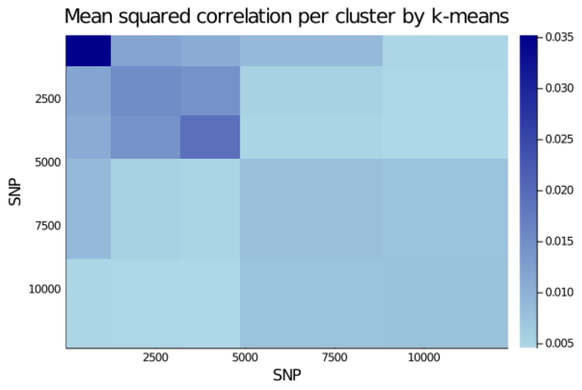


Figure 10.10: The average value of each cluster.

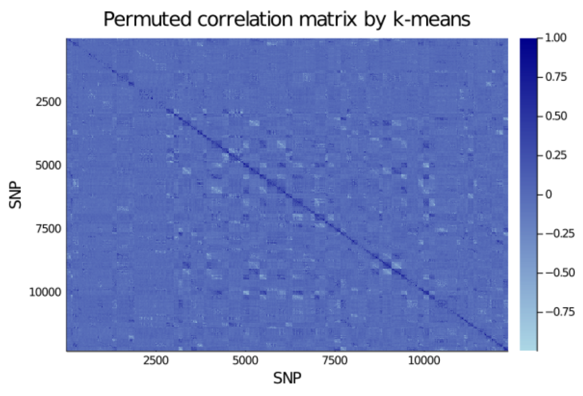


Figure 10.11: Correlation of the SNPs with 61 clusters.

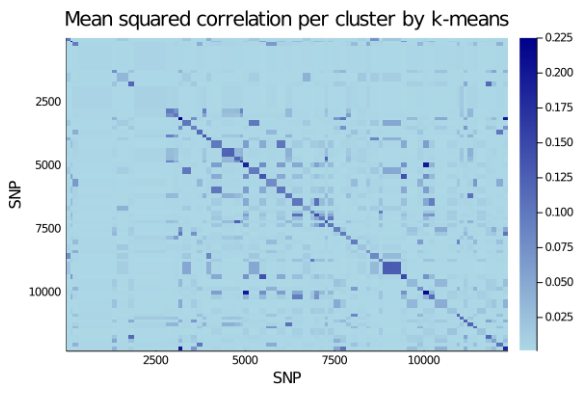


Figure 10.12: The average value of each cluster.

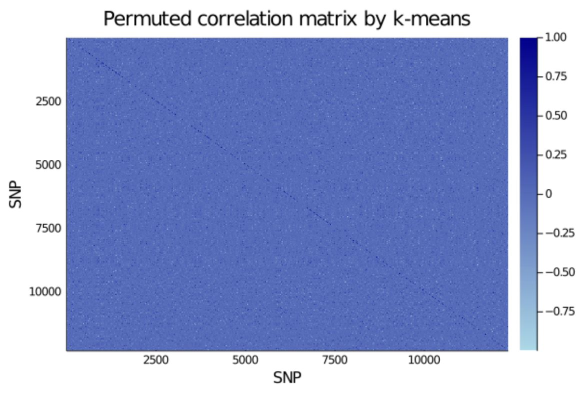


Figure 10.13: Correlation of the SNPs with 1234 clusters.

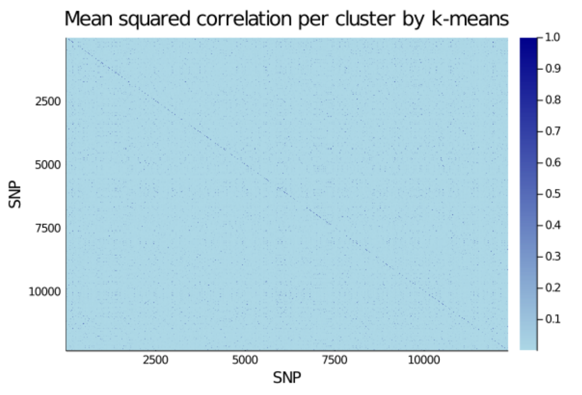


Figure 10.14: The average value of each cluster.

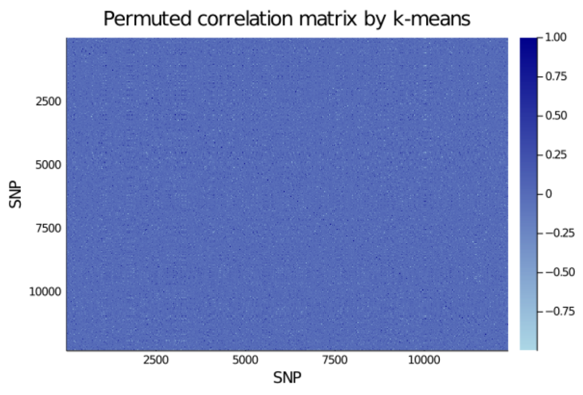


Figure 10.15: Correlation of the SNPs with 4115 clusters.

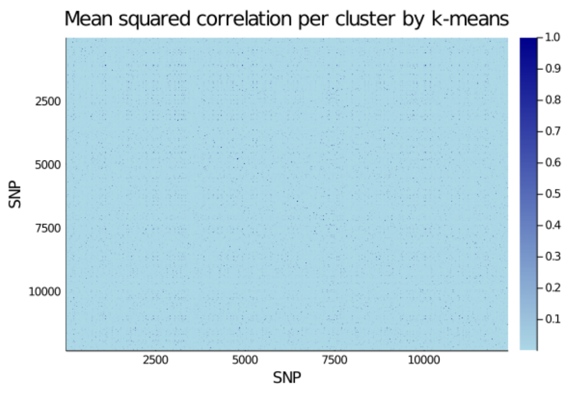


Figure 10.16: The average value of each cluster.

k-means algorithm equidistant

Results of the following number of clusters can be found here $\{5, 61, 123, 1234, 4115\}$ by applying the k-means algorithm using equidistant initial cluster centres.

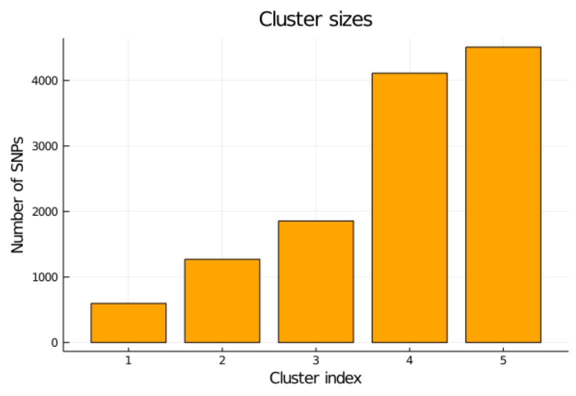


Figure 10.17: All cluster sizes with 5 cluster.

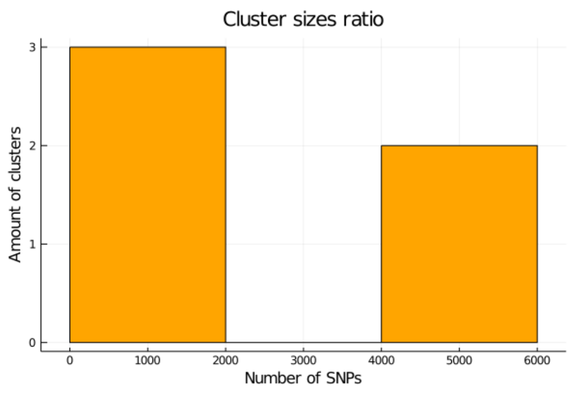


Figure 10.18: Histogram of the cluster sizes.

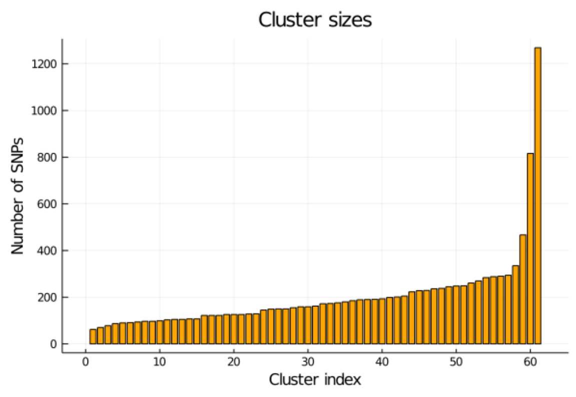


Figure 10.19: All cluster sizes with 61 clusters.

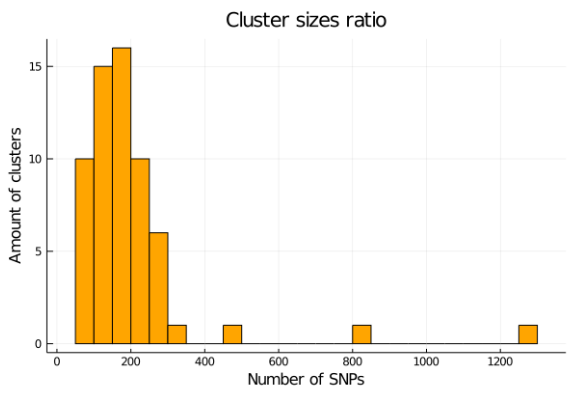


Figure 10.20: Histogram of the cluster sizes.

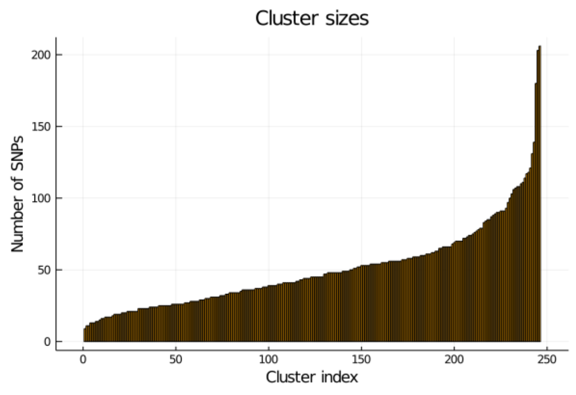


Figure 10.21: All cluster sizes with 123 clusters.

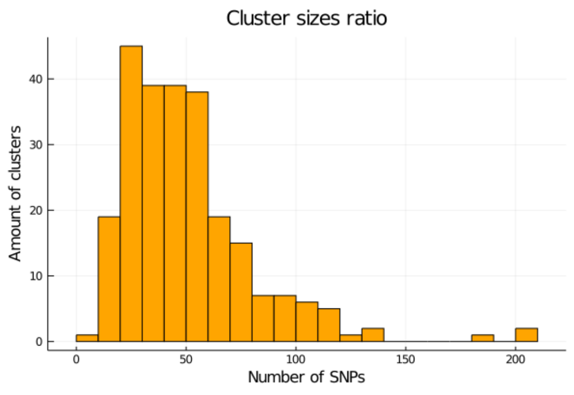


Figure 10.22: Histogram of the cluster sizes.

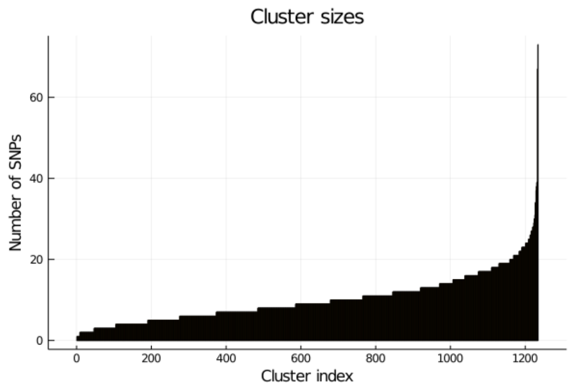


Figure 10.23: All cluster sizes with 1234 clusters.

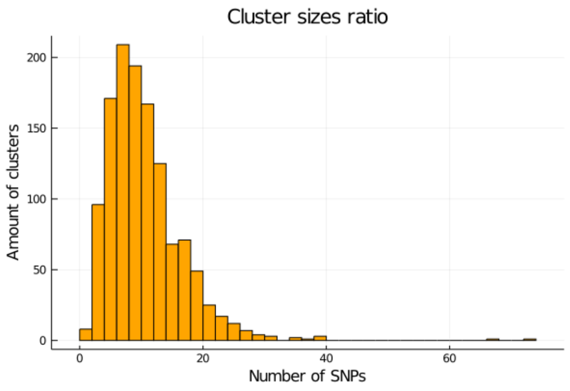


Figure 10.24: Histogram of the cluster sizes.

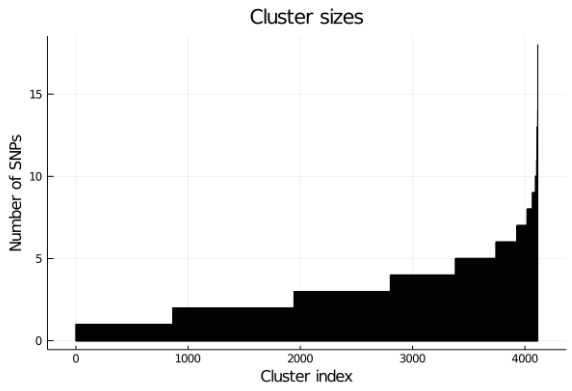


Figure 10.25: All cluster sizes with 4115 clusters.

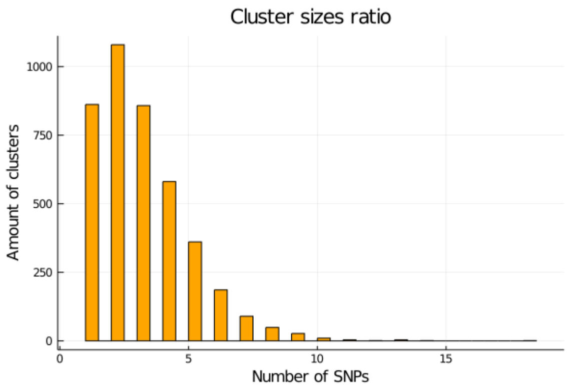


Figure 10.26: Histogram of the cluster sizes.

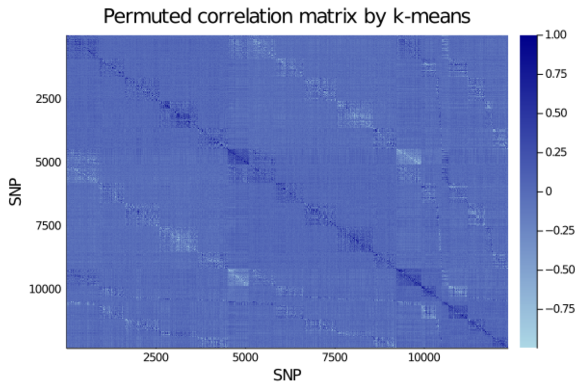


Figure 10.27: Correlation of the SNPs with 5 clusters.

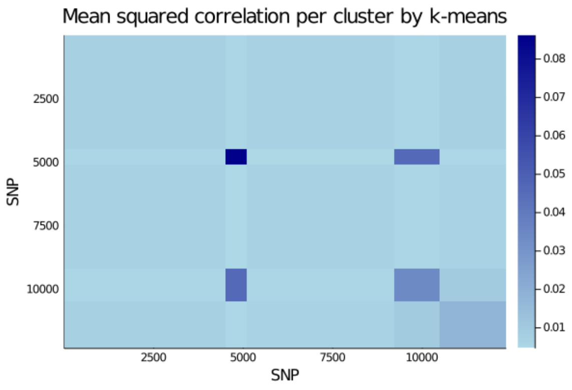


Figure 10.28: The average value of each cluster.

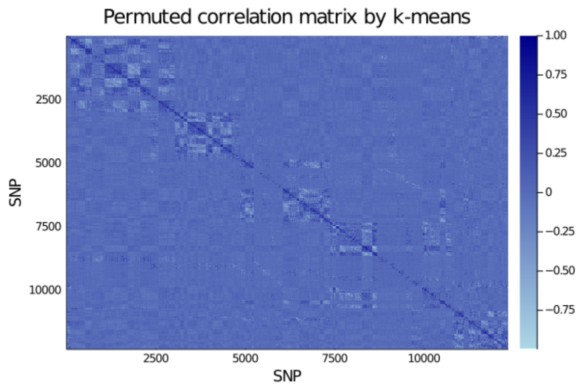


Figure 10.29: Correlation of the SNPs with 61 clusters.

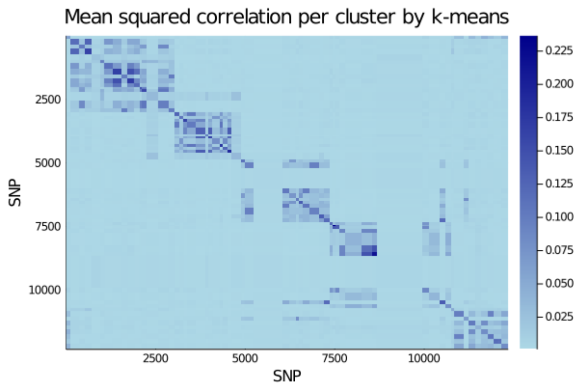


Figure 10.30: The average value of each cluster.

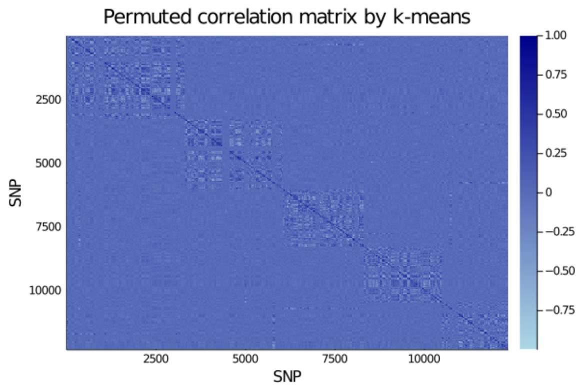


Figure 10.31: Correlation of the SNPs with 246 clusters.

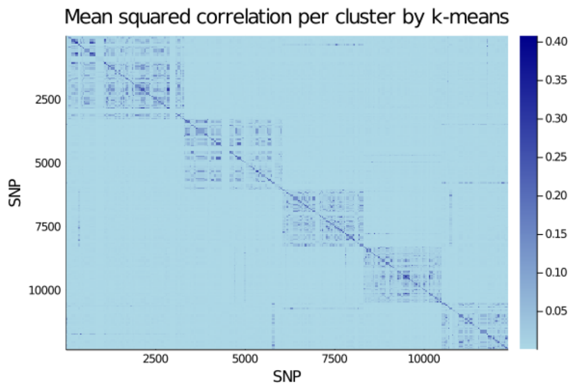


Figure 10.32: The average value of each cluster.

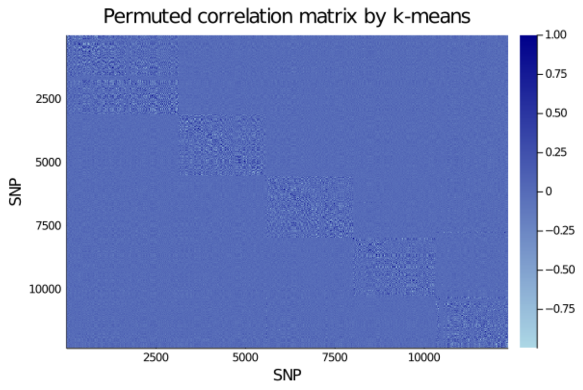


Figure 10.33: Correlation of the SNPs with 1234 clusters.

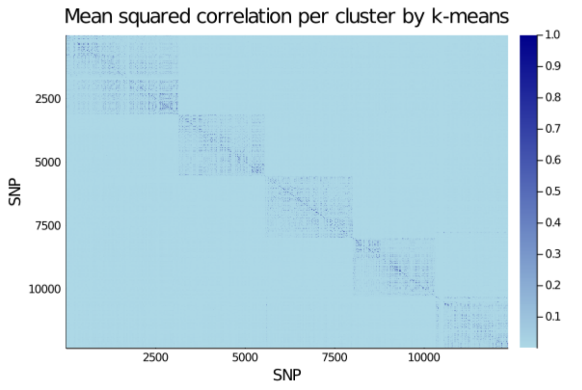


Figure 10.34: The average value of each cluster.

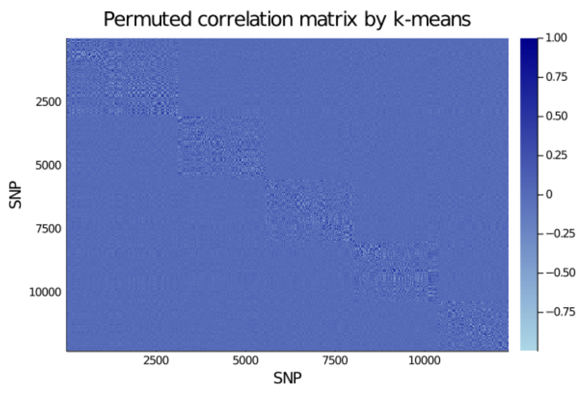


Figure 10.35: Correlation of the SNPs with 4115 clusters.

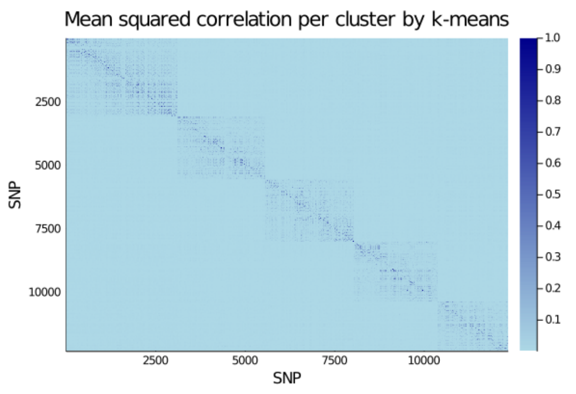


Figure 10.36: The average value of each cluster.

k-means++ algorithm

Results of the following number of clusters can be found here $\{5, 61, 123, 1234, 4115\}$ by applying the k-means++ algorithm .

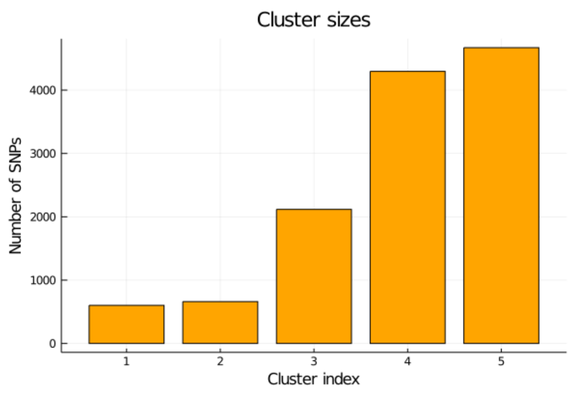


Figure 10.37: All cluster sizes with 5 cluster.

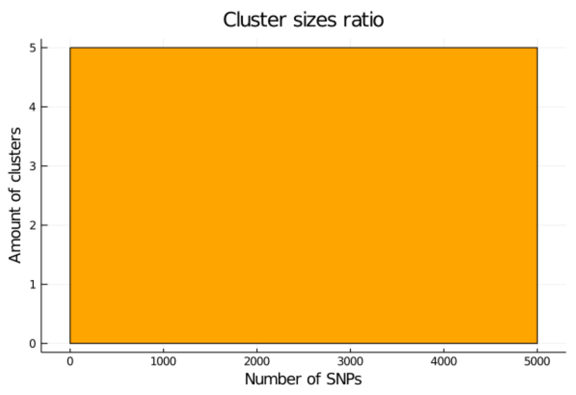


Figure 10.38: Histogram of the cluster sizes.

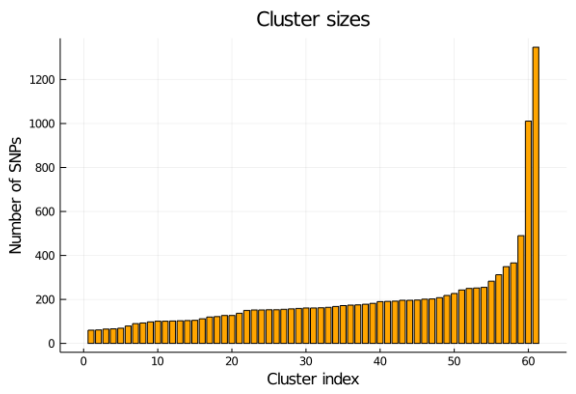


Figure 10.39: All cluster sizes with 61 clusters.

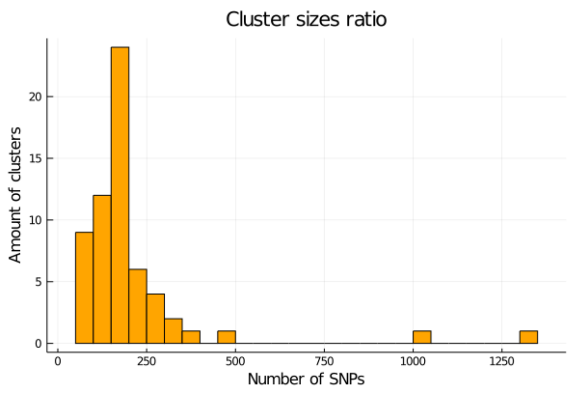


Figure 10.40: Histogram of the cluster sizes.

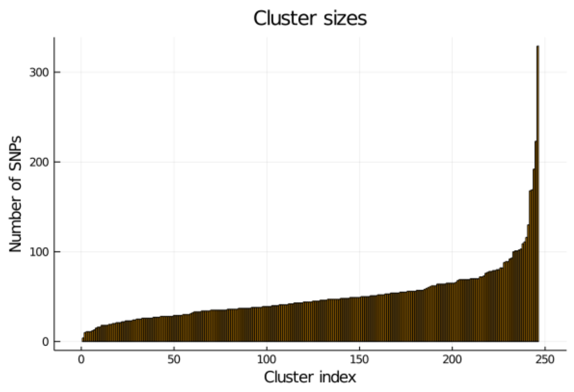


Figure 10.41: All cluster sizes with 123 clusters.

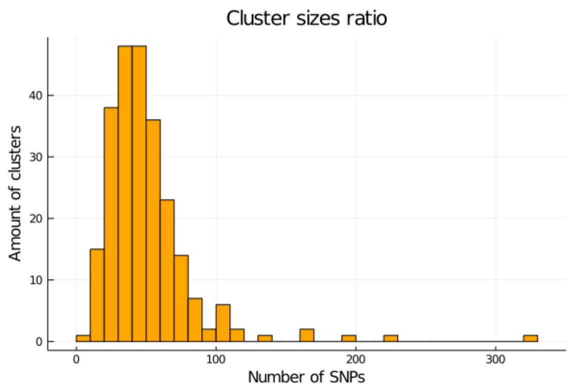


Figure 10.42: Histogram of the cluster sizes.

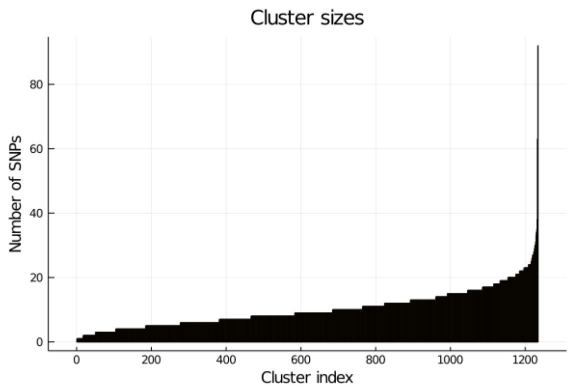


Figure 10.43: All cluster sizes with 1234 clusters.

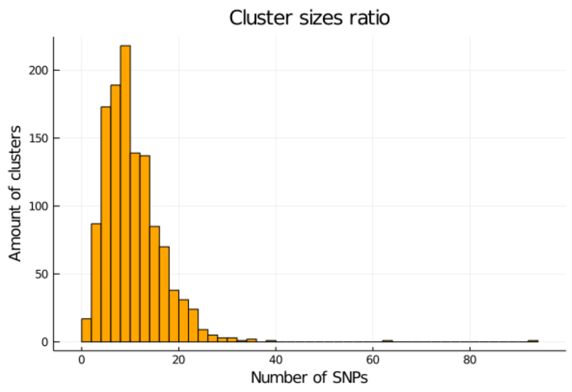


Figure 10.44: Histogram of the cluster sizes.

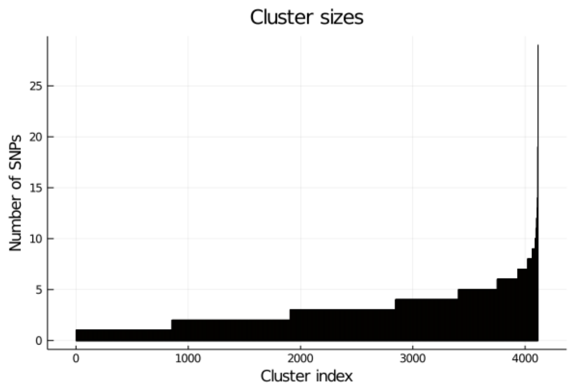


Figure 10.45: All cluster sizes with 4115 clusters.

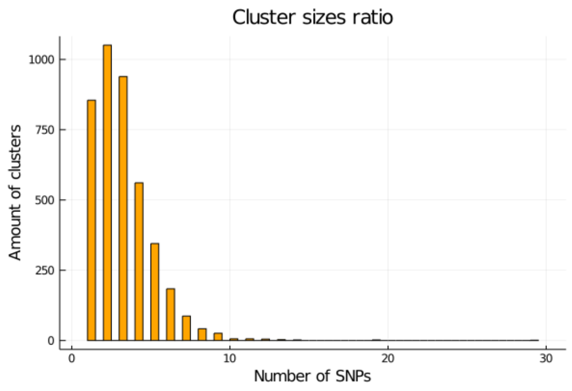


Figure 10.46: Histogram of the cluster sizes.

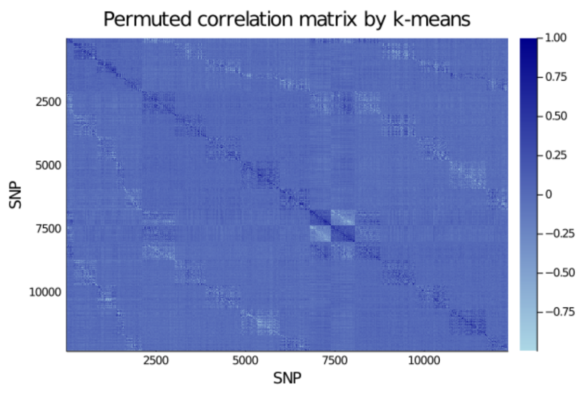


Figure 10.47: Correlation of the SNPs with 5 clusters.

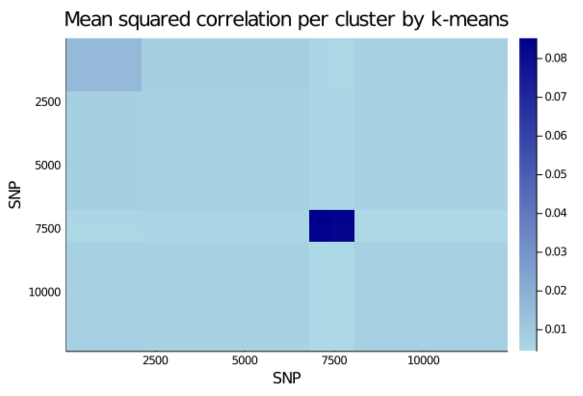


Figure 10.48: The average value of each cluster.

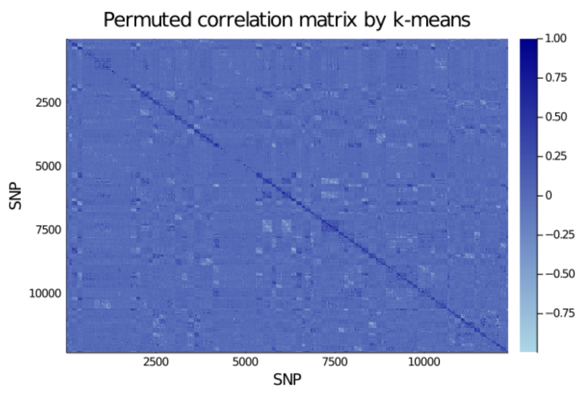


Figure 10.49: Correlation of the SNPs with 61 clusters.

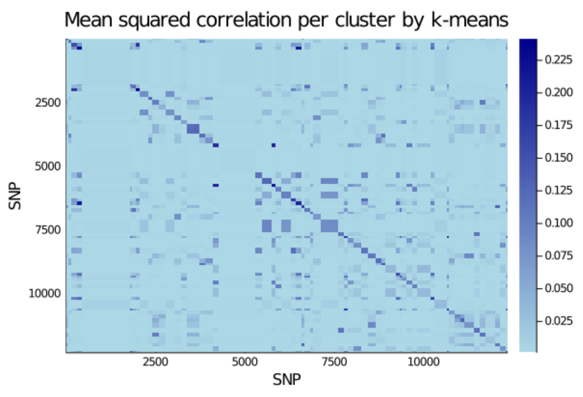


Figure 10.50: The average value of each cluster.

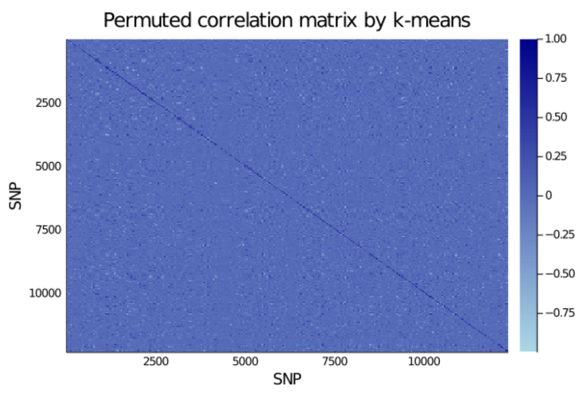


Figure 10.51: Correlation of the SNPs with 246 clusters.

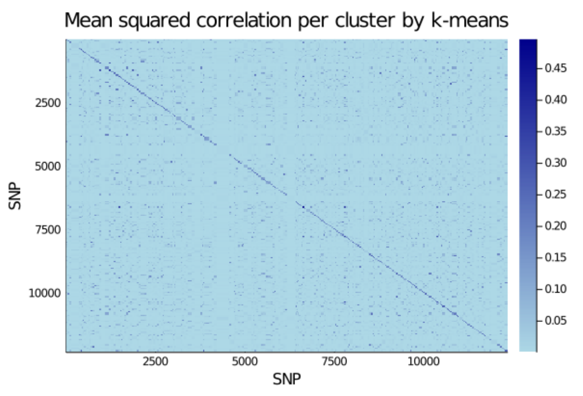


Figure 10.52: The average value of each cluster.

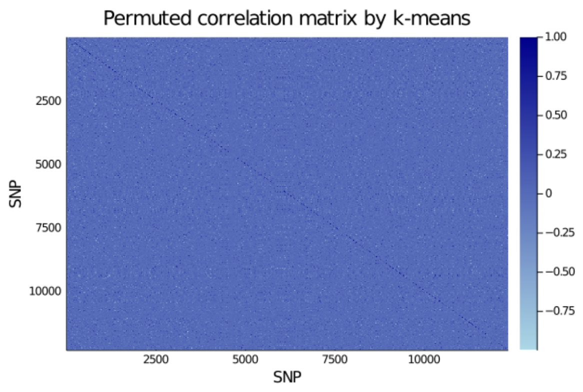


Figure 10.53: Correlation of the SNPs with 1234 clusters.

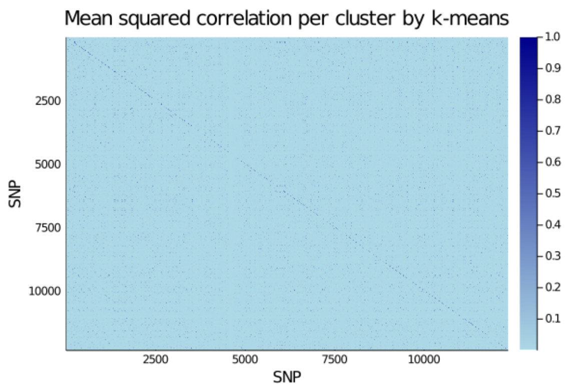


Figure 10.54: The average value of each cluster.

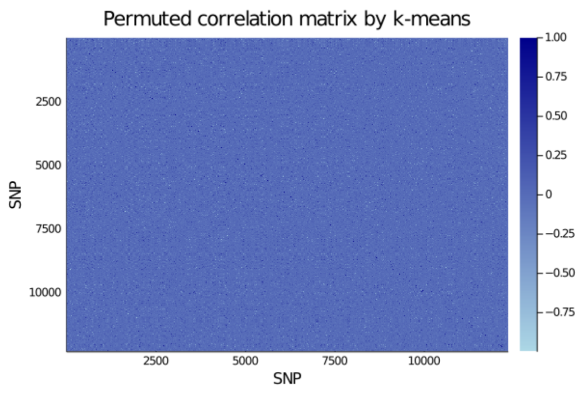


Figure 10.55: Correlation of the SNPs with 4115 clusters.

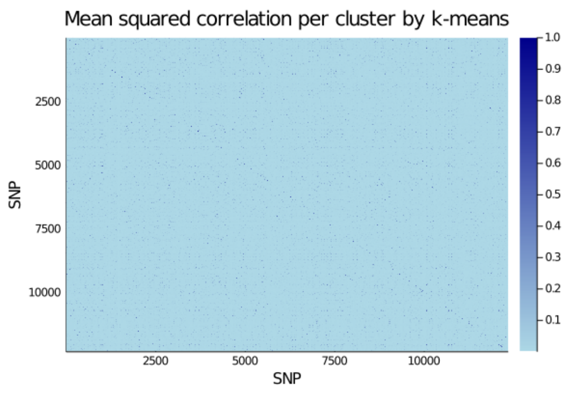


Figure 10.56: The average value of each cluster.

10.4 Julia set-up and modules

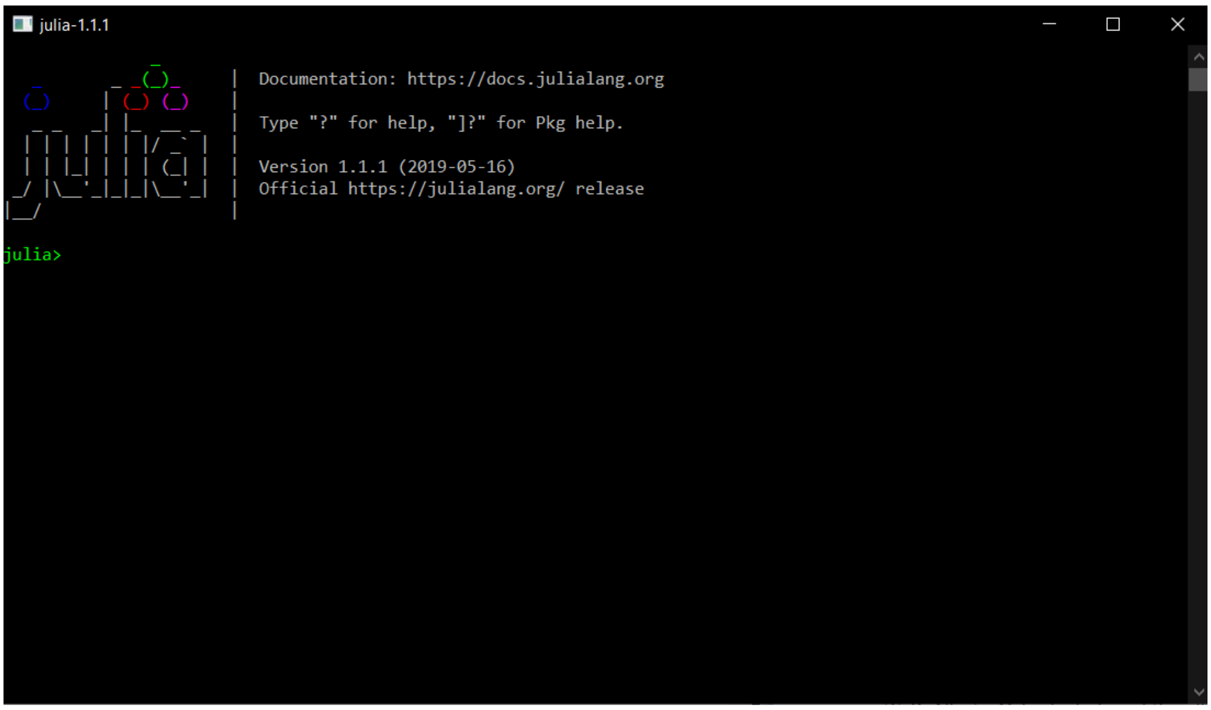
This chapter is dedicated to explain how to set-up Julia on a computer. It also includes a short summary of modules/libraries that are used for the experiments. For practical purposes Jupyter Notebook has been used for writing code snippets. Jupyter Notebook can be used as an IDE and provides a clean visual way to work on parts of your code.

Setting-up Julia

For the experiments Julia 1.2.0. has been utilised. At the moment writing version 1.6.0. is the newest stable version, older releases can be downloaded from https://julialang.org/downloads/oldreleases/.

Setting-up Jupyter Notebook for Julia

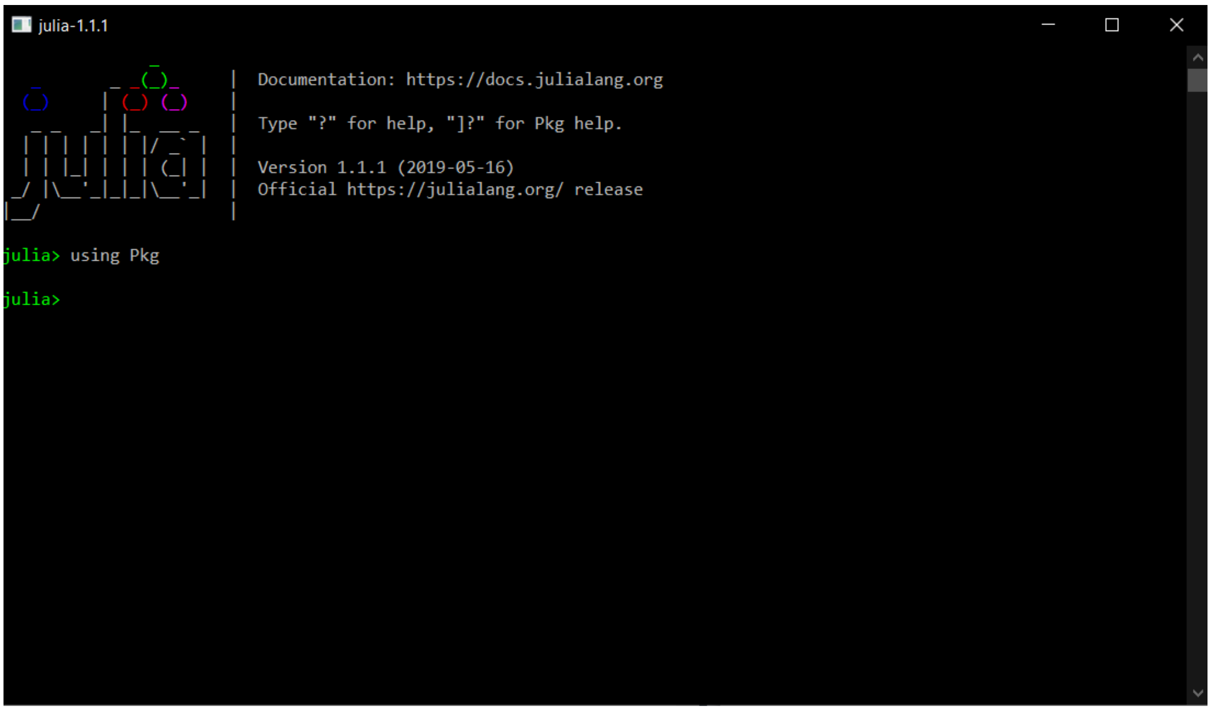
Once the proper version has been downloaded, proceed to open the Julia console, which should be similar to this:



```
julia-1.1.1

               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.1.1 (2019-05-16)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia>
```

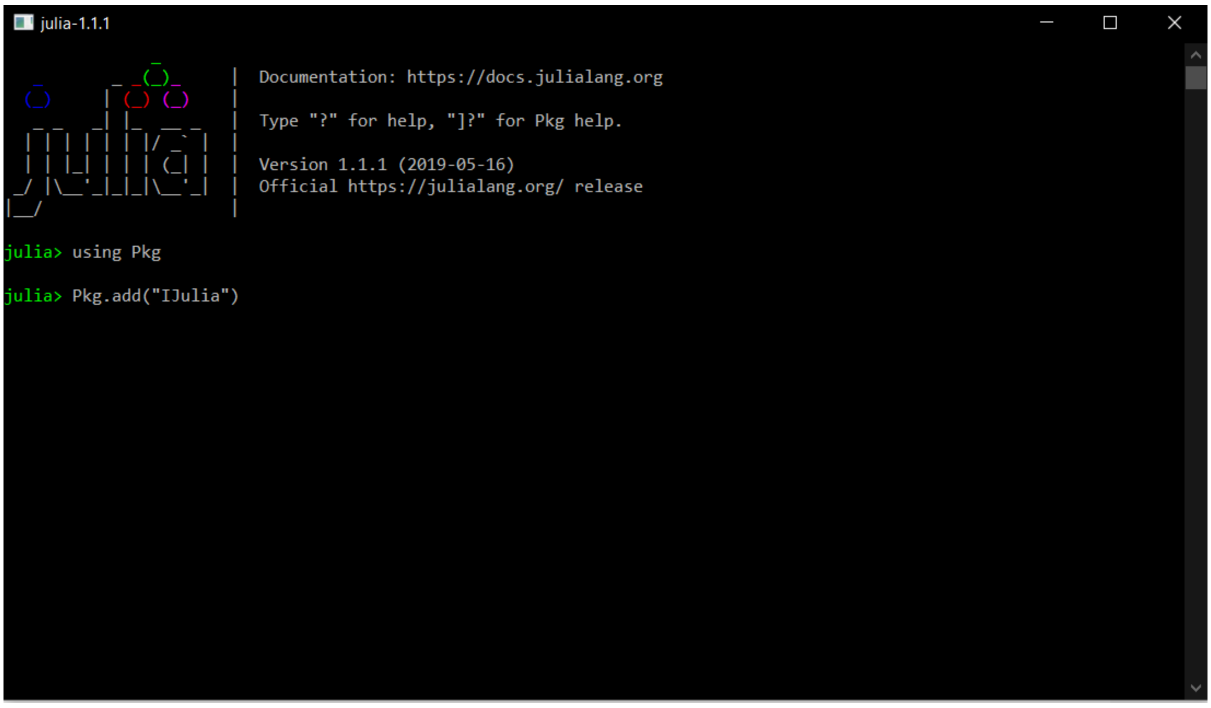
Figure 10.57: Julia console. Although version 1.1.1 is seen in the figure. The 1.2.0 version should look very similar.

Now in the console you can add modules/libraries that provide you with additional functions to work with. To add packages one wants to use the **Pkg** library. The library can be activated as follows:



```
julia-1.1.1

               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.1.1 (2019-05-16)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using Pkg

julia>
```

The **IJulia** library is required to let Julia work on Jupyter Notebook. It can be installed through the **Pkg** library, this library can be loaded by doing:

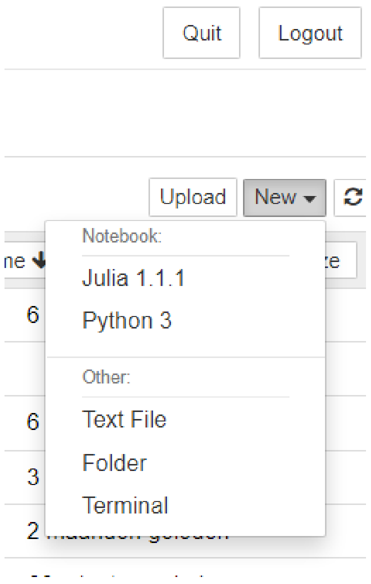


```
julia-1.1.1

               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.1.1 (2019-05-16)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> using Pkg

julia> Pkg.add("IJulia")
```

To install Jupyter Notebook, one can start downloading and installing Anaconda. Anaconda is an application that contains Python and R and it also includes Jupyter Notebook. The download link for Anaconda: https://www.anaconda.com/products/

individual.

To run Jupyter Notebook, one needs to have a browser application, but no connection with the internet is required. Once Anaconda has been installed, one can open Jupyter Notebook. To create a Jupyter Notebook file that can run Julia, one has to go through the following tab:



Clicking on **Julia <version>** will create a Jupyter Notebook file that can run Julia in Jupyter Notebook.

Required libraries for Julia

Here a list of libraries that haven been used for the experiments is shown.

Table 10.1: List of libraries.

Library	**Description**
Arpack	Includes FORTRAN routines for eigenvalues and eigenvectors estimates.
Clustering	Clustering algorithms.
DelimitedFiles	Read and write to files.
Glob	Search into directories.
LinearAlgebra	Fundamental matrix and vector operations.
Plots	Create plots.
SparseArrays	Fundamental functions for sparse matrices.
Random	Setting seed.
Statistics	Random sampling and other popular statistical functions.

List of modules to set-up the ssSNPBLUP model and DPCG algorithm.

Table 10.2: List of modules.

Module	**Description**
modblup	Functions for creating the ssSNPBLUP model.
modgeneral	Functions for showing data and reading genotype data files.
modpedigree	Functions to create the pedigree relationship matrix.
modreliabilities	Function for ssGBLUP.
modsolver	Functions for constructing the deflation matrix, running the PCG and DPCG algorithm.

10.5 FLOP count of CG, PCG, DCG and DPCG algorithm

The derivation of the computational cost of the iterative solver CG, PCG, DCG and DPCG can be found here. The computation cost is expressed in the total amount floating-point arithmetic operations, which are addition, subtraction, multiplication and division. This is usually denoted as floating-point operations (FLOP).

Define k, m and n be positive integers. The dimensions of the relevant matrices, vectors and parameters involved in the calculation of FLOPS are defined as:

$$\begin{aligned} A, C &\in \mathbb{R}^{n\times n} \\ B &\in \mathbb{R}^{n\times m} \\ p^k, r^k, w^k, x^k, b, x, y &\in \mathbb{R}^n \\ \alpha_k, \beta_k &\in \mathbb{R} \end{aligned}$$

Certain operations appear repeatedly in the chosen iterative solvers and have the following FLOP count:

Table 10.3: FLOP count of matrix and vector operations. It is assumed that A, B and C are dense.

Name	**Operator**	**FLOP**
Dot product	$< \cdot, \cdot >$	$2n - 1$
Vector addition/subtraction	$x + y$	n
Vector-scalar multiplication	αx	n
Matrix-vector multiplication	Ax	$n(2n - 1)$
Matrix multiplication	AB	$np(2n - 1)$
Matrix addition/subtraction	A + C	n^2

CG

For the derivations of FLOP, it will be assumed that A is a dense symmetric matrix. Thus the following results will be an estimate of the upper bound of FLOP. Consider the linear system:

$$Ax = b$$

Then the FLOP count of the CG algorithm is given in table 10.4.

Table 10.4: FLOP count of the CG method.

	Operation	**FLOP**
Initialisation	Choose x^0	0
	$r^0 = b - Ax^0$	$2n^2$
	$p^0 = r^0$	0
	FLOP initialisation	$\mathbf{2n^2}$
Iteration	$w^k = Ap^k$	$n(2n-1)$
	$\alpha_k = \frac{<r^k, r^k>}{<p^k, w^k>}$	$2(2n-1)+1$
	$x^{k+1} = x^k + \alpha_k p^k$	$2n$
	$r^{k+1} = r^k - \alpha_k w^k$	$2n$
	$\beta_k = \frac{<r^{k+1}, r^{k+1}>}{<r^k, r^k>}$	$2(2n-1)+1$
	$p^{k+1} = r^{k+1} + \beta_k p^k$	$2n$
	FLOP iteration	$\mathbf{2n^2 + 13n - 2}$

PCG

Assuming that the preconditioner matrix $M \in \mathbb{R}^{n \times n}$ is a diagonal matrix. It has n non-zero elements which are positioned on the main diagonal. Then the following holds:

Table 10.5: FLOP count of matrix operations where M is a diagonal matrix and A is a dense matrix.

Name	**Operator**	**FLOP**
Matrix-vector multiplication	Mx	n
Matrix multiplication	MA	n^2
Matrix addition/subtraction	M + A	n

Table 10.6: FLOP count of the PCG method.

	Operation	**FLOP**
Initialisation	Choose x^0	0
	$r^0 = b - Ax^0$	$2n^2$
	$y^0 = M^{-1} r^0$	n
	$p^0 = y^0$	0
	FLOP initialisation	$\mathbf{2n^2 + n}$
Iteration	$w^k = Ap^k$	$n(2n-1)$
	$\alpha_k = \frac{<r^k, y^k>}{<p^k, w^k>}$	$2(2n-1)+1$
	$x^{k+1} = x^k + \alpha_k p^k$	$2n$
	$r^{k+1} = r^k - \alpha_k w^k$	$2n$
	$y^{k+1} = M^{-1} r^{k+1}$	n
	$\beta_k = \frac{<r^{k+1}, y^{k+1}>}{<r^k, y^k>}$	$2(2n-1)+1$
	$p^{k+1} = y^{k+1} + \beta_k p^k$	$2n$
	FLOP iteration	$\mathbf{2n^2 + 14n - 2}$

Comparing the FLOP count with the CG method, the additional cost is due to the matrix-vector multiplication $y^{k+1} = M^{-1}r^{k+1}$, which is n additional FLOP.

DCG

Define $Z \in \mathbb{R}^{n\times m}$ as the deflation-subspace matrix with $m \geq 1$. If $m = 0$, then the DCG algorithm is equivalent to the CG method. Define the Galerkin matrix as $E = Z^TAZ$ and the deflation matrix $P = I - AZE^{-1}Z^T$. The previous assumptions are taken for all other similar matrices and vectors applied in the DCG algorithm. The matrix-vector multiplication with P can be done in 4 parts without P in memory. Assume that E^{-1} is in memory.

$$\begin{aligned} Px &= (I - AZE^{-1}Z^T)x \\ &= x - AZE^{-1}Z^Tx \end{aligned}$$

where $AZE^{-1}Z^Tx$ can be split up in 3 parts:

$$\begin{aligned} y &= Z^Tx \\ s &= E^{-1}y \\ z &= AZs \end{aligned}$$

Assuming that the matrix AZ has been computed and is saved in memory, then this results in 3 matrix-vector multiplications and 1 vector subtraction. Therefore, the computational cost of Px is $\mathbf{m(2n-1) + m(2m-1) + n(2m-1) + n = 4mn + 2m(m-1)}$. Analogously for PAx, the following can be obtained:

$$PAx = Ax - AZE^{-1}(AZ)^Tx$$

where $AZE^{-1}(AZ)^T$ can be split into 3 parts:

$$\begin{aligned} y &= (AZ)^Tx \\ s &= E^{-1}y \\ z &= AZs \end{aligned}$$

Yielding 4 matrix-vector multiplications and 1 vector subtraction. The computational cost of PAx is

$$\mathbf{n(2n-1) + m(2n-1) + m(2m-1) + n(2m-1) + n = 2n^2 + (4m-1)n + 2m(m-1)}$$

To obtain a solution x of the linear system $Ax = b$. The solution $\hat{x}^k$ obtained from the iteration process has to be adjusted by:

$$x = Qb + P^T\hat{x}^{k+1}$$

This can be reformulated as:

$$\begin{aligned} x &= Qb + (I - QA)^T\hat{x} \\ &= \hat{x} + Q(b - A\hat{x}) \\ &= \hat{x} + ZE^{-1}(Z^Tb - (AZ)^T\hat{x}) \end{aligned}$$

Note that $Q = ZE^{-1}Z^T$ is symmetric. This computation consists of 4 matrix-vector multiplications and 2 vector additions/subtractions. The computational cost is given by:

$$
\begin{aligned}
&m + n + m(2n-1) + m(2n-1) + m(2m-1) + n(2m-1) = \\
&m + n + 2m(2n-1) + (m+n)(2m-1) = \\
&2m(2n-1) + (m+n)2m = \\
&6mn + 2m(m-1)
\end{aligned}
$$

Comparing these alternative ways of computations with a deflation matrix P and correction matrix Q in memory and disregarding the computational cost of constructing both matrices. Let $n \geq 1$. The following can be obtained.

- Px has $n(2n-1)$ FLOP and the alternative method has $4mn+2m(m-1)$ FLOP. To derive the dimension m that the deflation-subspace matrix can take on such that the alternative method is cheaper or equal in computational cost, the following inequality has to be solved for m:

$$
\begin{aligned}
4mn + 2m(m-1) &\leq n(2n-1) \\
2m^2 + (4n-2)m - n(2n-1) &\leq 0
\end{aligned}
$$

  Solving this yields:

$$m \leq \frac{\sqrt{32n^2 - 16n + 4} - 4n + 2}{4}$$

- PAx has $2n(2n-1)$ FLOP and the alternative method has $2n^2 + 4(m-1)n + 2m(m-1)$ FLOP. Consider the following inequality:

$$
\begin{aligned}
2n^2 + 4(m-1)n + 2m(m-1) &\leq 2n(2n-1) \\
2m^2 + (4n-2) + 2n^2 - n - 2n(n-1) &\leq 0
\end{aligned}
$$

  Solving this yields:

$$m \leq \frac{\sqrt{48n^2 - 16n + 4} - 4n + 2}{4}$$

- $x = Qb + P^T\hat{x}$ has $2n(2n-1) + n$ FLOP and the alternative method has $6mn + 2m(m-1)$ FLOP. Consider the following inequality:

$$
\begin{aligned}
6mn + 2m(m-1) &\leq 2n(2n-1) + n \\
2m^2 + (6n-2)m - n(4n-1) &\leq 0
\end{aligned}
$$

  Solving this yields:

$$m \leq \frac{\sqrt{68n^2 - 20n + 4} - 6n + 2}{4}$$

Choose $m \in \mathbb{N}_{n\geq 1}$, then the DCG method is computationally cheaper using the alternative methods, if m satisfies the smallest upper bound. Therefore, the following must hold:

$$m \leq \frac{\sqrt{32n^2 - 16n + 4} - 4n + 2}{4} \approx \frac{\sqrt{32n^2} - 4n}{4} = (\sqrt{2} - 1)n$$

where the latter approximation holds for $n \gg 1$. Hence, the deflation subspace matrix can have at most $\sim 40\%$ of the amount of rows in A as columns.

Table 10.7: FLOP count of the DCG method.

	Operation	**FLOP**
Initialisation	Choose x^0	0
	Compute AZ	$mn(2n-1)$
	$r^0 = b - Ax^0$	$2n^2$
	$\hat{r}^0 = Pr^0$	$4mn + 2m(m-1)$
	$p^0 = \hat{r}^0$	0
	FLOP initialisation	$\mathbf{2(m+1)n^2 + 4mn + m(2m-3)}$
Iteration	$\hat{w}^k = PAp^k$	$2n^2 + 4(m-1)n + 2m(m-1)$
	$\alpha_k = \frac{<\hat{r}^k,\hat{r}^k>}{<p^k,\hat{w}^k>}$	$2(2n-1)+1$
	$\hat{x}^{k+1} = \hat{x}^k + \alpha_k p^k$	$2n$
	$\hat{r}^{k+1} = \hat{r}^k - \alpha_k \hat{w}^k$	$2n$
	$\beta_k = \frac{<\hat{r}^{k+1},\hat{r}^{k+1}>}{<\hat{r}^k,\hat{r}^k>}$	$2(2n-1)+1$
	$p^{k+1} = \hat{r}^{k+1} + \beta_k p^k$	$2n$
	FLOP iteration	$\mathbf{2n^2 + (4m+10)n + 2m(m-1) - 2}$
Final	$\mathbf{x_{final} = Qb + P^T\hat{x}^{k+1}}$	$\mathbf{6mn + 2m(m-1)}$

The additional cost compared with the CG method is attributed to the matrix-vector and matrix-matrix multiplication with the deflation matrix P. The difference in FLOP for the initialisation process compared with the CG method is $2mn^2 + 4mn + m(2m-3)$. For the iteration process the difference is $(4m-3)n + 2m(m-1)$.

DPCG

The previous assumptions for the relevant matrices and vectors also hold here.

Table 10.8: FLOP count of the DPCG method.

	Operation	**FLOP**
Initialisation	Choose x^0	0
	Compute AZ	$mn(2n-1)$
	$r^0 = b - Ax^0$	$2n^2$
	$\hat{r}^0 = Pr^0$	$4mn + 2m(m-1)$
	$y^0 = M^{-1}\hat{r}^0$	n
	$p^0 = y^0$	0
	FLOP initialisation	$\mathbf{2(m+1)n^2 + 4(m+1)n + m(2m-3)}$
Iteration	$\hat{w}^k = PAp^k$	$2n^2 + 4(m-1)n + 2m(m-1)$
	$\alpha_k = \frac{<\hat{r}^k, y^k>}{<p^k, \hat{w}^k>}$	$2(2n-1)+1$
	$\hat{x}^{k+1} = \hat{x}^k + \alpha_k p^k$	$2n$
	$\hat{r}^{k+1} = \hat{r}^k - \alpha_k \hat{w}^k$	$2n$
	$y^{k+1} = M^{-1}\hat{r}^{k+1}$	n
	$\beta_k = \frac{<\hat{r}^{k+1}, y^{k+1}>}{<\hat{r}^k, y^k>}$	$2(2n-1)+1$
	$p^{k+1} = y^{k+1} + \beta_k p^k$	$2n$
	FLOP iteration	$\mathbf{2n^2 + (4m+11)n + 2m(m-1) - 2}$
Final	$\mathbf{x_{final} = Qb + P^T\hat{x}^{k+1}}$	$\mathbf{6mn + 2m(m-1)}$

The additional cost compared with the CG method is attributed to the matrix-vector and matrix-matrix multiplication with the deflation matrix P and the matrix-vector multiplication with the inverse of the preconditioner matrix M^{-1}. The difference in FLOP for the initialisation process compared with the CG method is $2mn^2 + (4m+1)n + m(2m-3)$. For the iteration process the difference is $4(m-1)n + 2m(m-1)$.

Sparse matrix

Assume that the matrix $A \in \mathbb{R}^{n \times n}$ is sparse with $q \in \mathbb{N}$ nonzero elements. Define $m \in \mathbb{N}_{\geq 1}$ as the number of deflation vectors. Define $B \in \mathbb{R}^{n \times k}$ and $x \in \mathbb{R}^n$. The matrix-vector and matrix-matrix multiplication with A have the following FLOP count. For the latter case the other matrix is assumed to be dense.

Table 10.9: FLOP count of matrix and vector operations. It is assumed that A is sparse and B is dense.

Name	Operator	FLOP
Matrix-vector multiplication	Ax	$2q-1$
Matrix multiplication	AB	$k(2q-1)$

The CG, PCG, DCG and DPCG method have the following FLOP count on initialisation and iteration.

Table 10.10: FLOP count of the CG, PCG, DCG and DPCG method. The variable n denotes the size of the square coefficient matrix A, q denotes the amount of nonzero elements matrix A has and m is the number of deflation vectors.

	Initialisation	**Iteration**
CG	$n + 2q - 1$	$14n + 2q - 3$
PCG	$2n + 2q - 1$	$15n + 2q - 3$
DCG	$(4m + 1)n + m(2m - 2q - 3) + 2q - 1$	$(4m + 14)n + 2m(m - 1) + 2q - 3$
DPCG	$(4m + 2)n + m(2m - 2q - 3) + 2q - 1$	$(4m + 15)n + 2m(m - 1) + 2q - 3$

Inverse of the Galerkin matrix

Define $m \in \mathbb{N}_{\geq 1}$ as the number of deflation vectors. If the computation of the inverse of the Galerkin matrix E is computationally expensive, then a direct solver or iterative solver can be opted. For a dense Galerkin matrix it takes more FLOP to calculate the inverse directly than solving it with a forward and backward substitution of a Cholesky decomposition [7].
The method to apply a direct solver will be explained. Define $r, s \in \mathbb{R}^m$. The linear system for solving s is given by:

$$s = E^{-1}r$$

which can be rewritten as

$$Es = r$$

Assuming that $A \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix, then the Galerkin Matrix E is by definition a symmetric positive semidefinite matrix. The Cholesky decomposition method can be applied on E. Obtaining the following equality $E = LL^T$ where L is a lower triangular matrix. Hence a forward substitution and then a backward substitution can be applied to solve this linear system.
The Cholesky decomposition has a computational complexity of $\mathcal{O}(n^3)$. The FLOP count of this method is $\mathbf{\frac{n^3}{3}}$ [7]. The forward and backward substitution methods have a computational complexity of $\mathcal{O}(n^2)$. Each of these methods have a FLOP count of $\mathbf{n^2}$ [7]. The Cholesky decomposition can be calculated once. Therefore, the forward and backward substitution will be the only recurring method per iteration in a CG algorithm. Thus the computational complexity per iteration is $\mathcal{O}(n^2)$ and the FLOP count is $\mathbf{2n^2}$.

Assume that the Cholesky decomposition has been performed and A is sparse with q nonzero elements. The following results hold for the interaction with the deflation matrix P:

Table 10.11: FLOP count with the Cholesky decomposition of the Galerkin matrix E.

Operator	**FLOP**
Px	$4mn + 2m(m - \frac{1}{2})$
PAx	$2n^2 + (4m - 1)n + 2m(m - \frac{1}{2})$

The FLOP count of the CG, PCG, DCG and DPCG algorithms where the Cholesky decomposition has been applied.

Table 10.12: FLOP count of the CG, PCG, DCG and DPCG method. The variable n denotes the size of the square coefficient matrix A, q denotes the amount of nonzero elements matrix A has and m is the number of deflation vectors.

	Initialisation	**Iteration**
CG	$n + 2q - 1$	$14n + 2q - 3$
PCG	$2n + 2q - 1$	$15n + 2q - 3$
DCG	$(4m + 1)n + m(2m - 2q - 2) + 2q - 1$	$(4m + 14)n + 2m(m - \frac{1}{2}) + 2q - 3$
DPCG	$(4m + 2)n + m(2m - 2q - 2) + 2q - 1$	$(4m + 15)n + 2m(m - \frac{1}{2}) + 2q - 3$

PCG vs DPCG algorithm cost

In this thesis the PCG algorithm is the baseline. If one wants to use the DPCG algorithm, then it should have less FLOP counts during the whole solve process than the PCG algorithm. Assume that the algorithms need the same amount of iterations for convergence, then it can be easily seen that the PCG algorithm is cheaper than the DPCG algorithm. However, in this research it has been observed that the DPCG algorithm takes less iterations for convergence. The additional work that the DPCG algorithm has to do compared with the PCG algorithm depends on the amount of deflation vectors used.

If there are k deflation vectors, what is the maximum amount of iterations required such that the DPCG algorithm is cheaper than the PCG algorithm?

The matrix studied in this research has dimensions 176847×176847. The matrix has 910472887 non-zero elements. Thus approximately 2.911% elements are non-zero elements. Therefore, this matrix can be considered sparse.
For the calculations of the ratio of FLOP counts between the PCG and DPCG method only the iteration process will be considered, since the initialisation only happens once. Calculating the FLOP count ratio between the PCG and DPCG method can be done by dividing the DPCG iteration FLOP count with the PCG iteration FLOP count, the FLOP counts can be seen in table 10.12. Performing this calculation, results in figure 10.58.

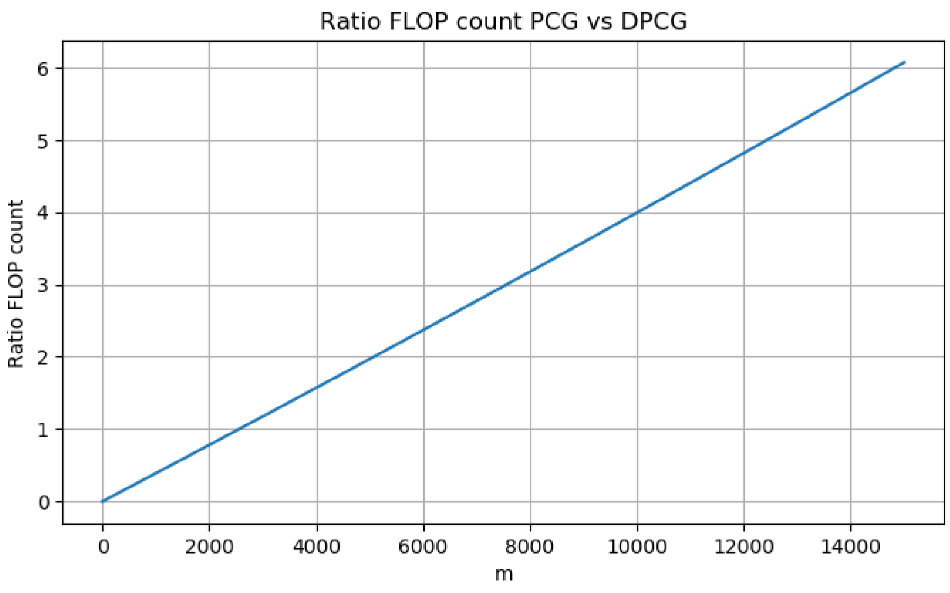


Figure 10.58: The FLOP count ratio between the PCG and DPCG algorithm. On the x-axis the variable m denotes the number of deflation vectors.

Figure 10.59 can be obtained by computing the reciprocal of one plus the FLOP count ratio.

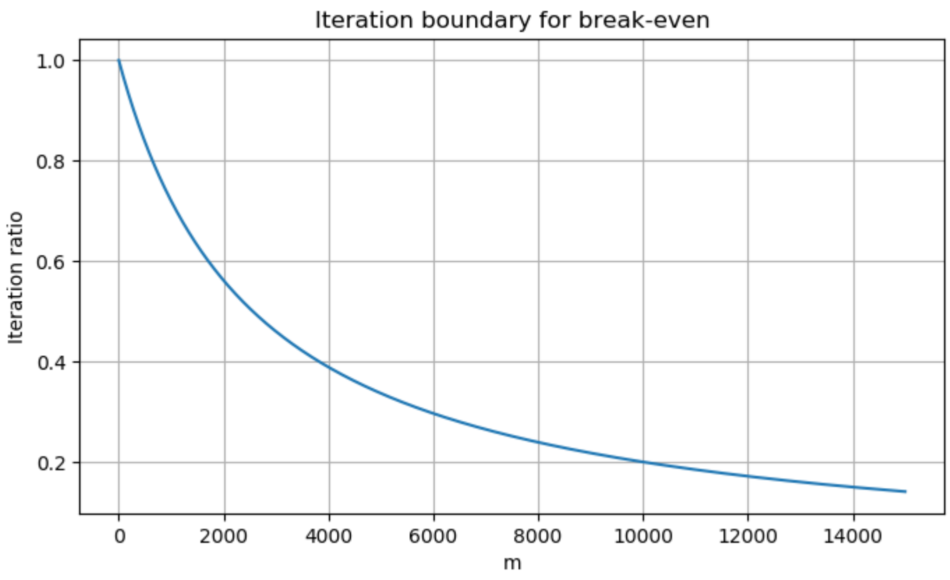


Figure 10.59: A threshold of iterations. On the x-axis the variable m denotes the number of deflation vectors. If the amount of iterations required for convergence of the DPCG algorithm is a value on top of the graph, then the DPCG algorithm is more expensive in FLOP count than the PCG algorithm.

10.6 Supplementary POD based deflation method results

The solutions of the reduced original system with reductions $\{0.5\%, 0.75\%, 1\%\}$.

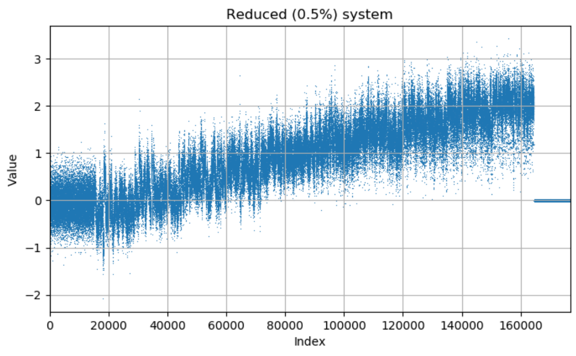


Figure 10.60: The solution of the reduced system with a reduction of 0.5%.

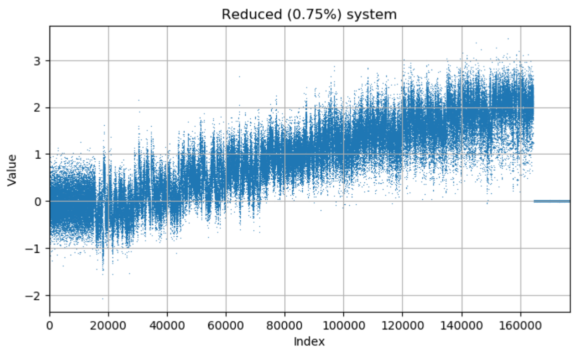


Figure 10.61: The solution of the reduced system with a reduction of 0.75%.

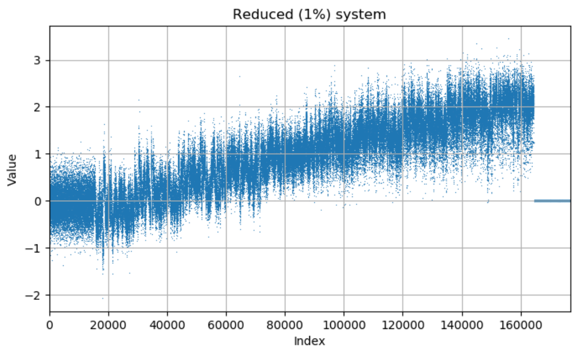


Figure 10.62: The solution of the reduced system with a reduction of 1.0%.

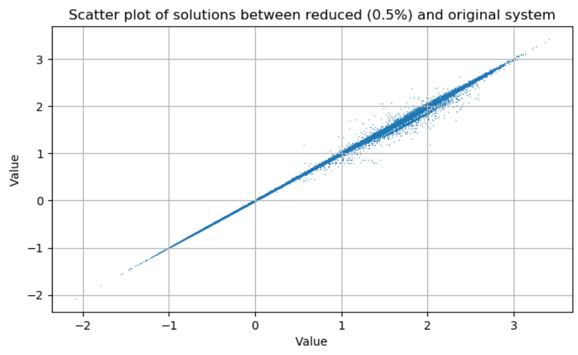


Figure 10.63: Comparison of values between solution of reduced and original system with a reduction of 0.5%.

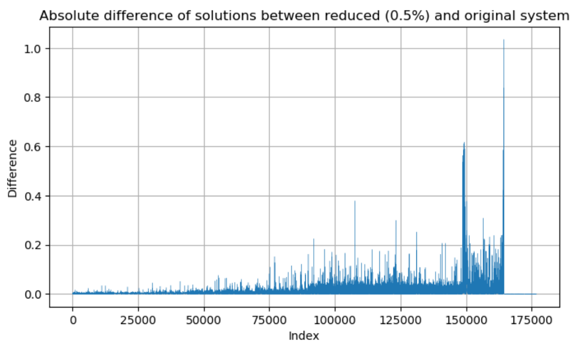


Figure 10.64: Absolute difference component wise of the solution of the reduced and original system with a reduction of 0.5%.

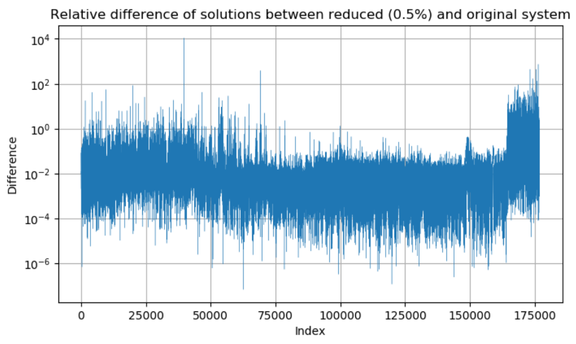


Figure 10.65: Relative difference component wise of the solution of the reduced and original system with a reduction of 0.5%.

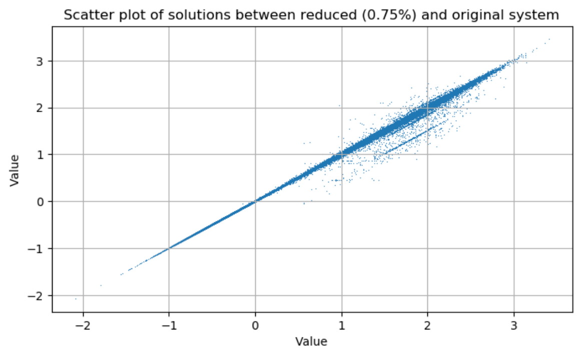


Figure 10.66: Comparison of values between solution of reduced and original system with a reduction of 0.75%.

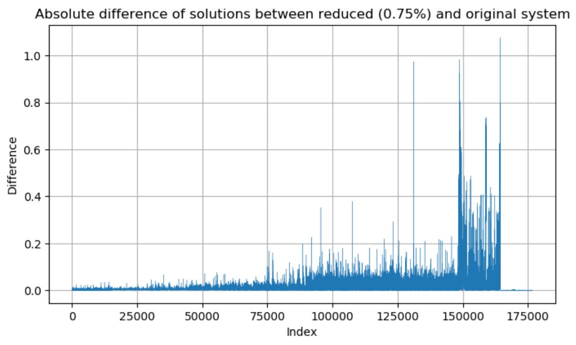


Figure 10.67: Absolute difference component wise of the solution of the reduced and original system with a reduction of 0.75%.

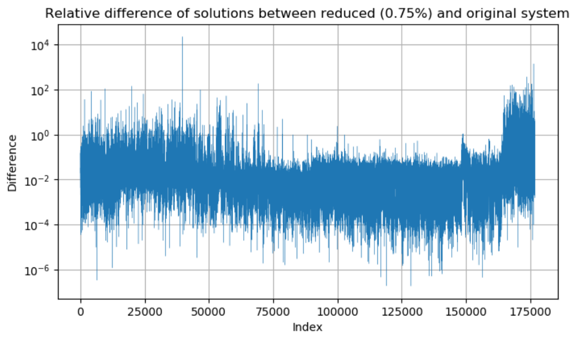


Figure 10.68: Relative difference component wise of the solution of the reduced and original system with a reduction of 0.75%.

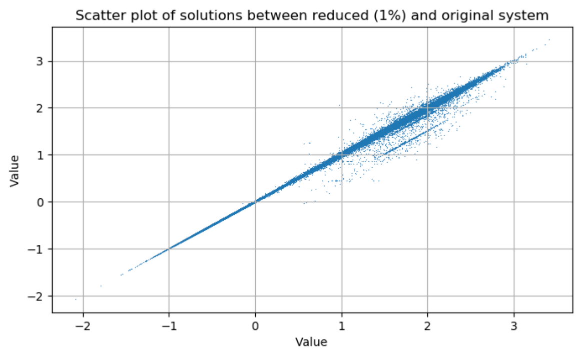


Figure 10.69: Comparison of values between solution of reduced and original system with a reduction of 1%.

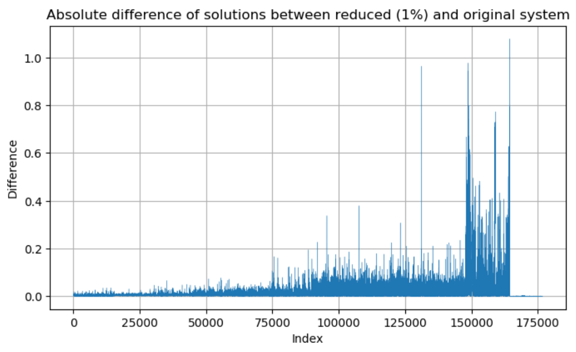


Figure 10.70: Absolute difference component wise of the solution of the reduced and original system with a reduction of 1%.

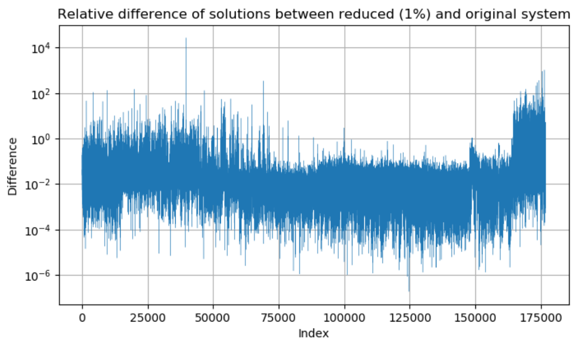


Figure 10.71: Relative difference component wise of the solution of the reduced and original system with a reduction of 1%.

The results of the POD method applied on the correlation matrix.

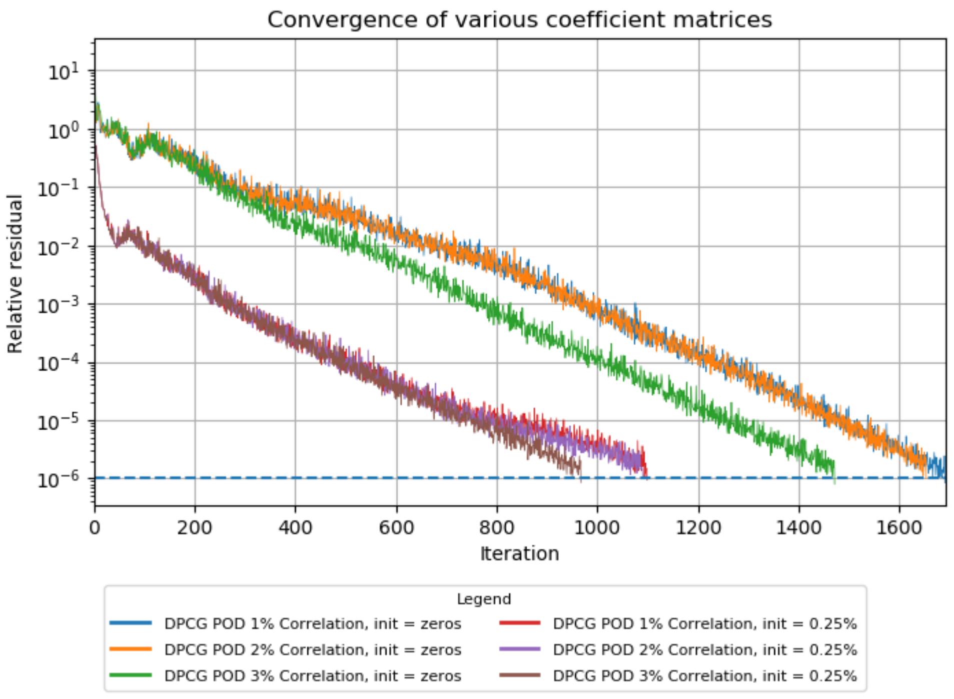


Figure 10.72: Comparison between different POD basis and two different initial solutions.

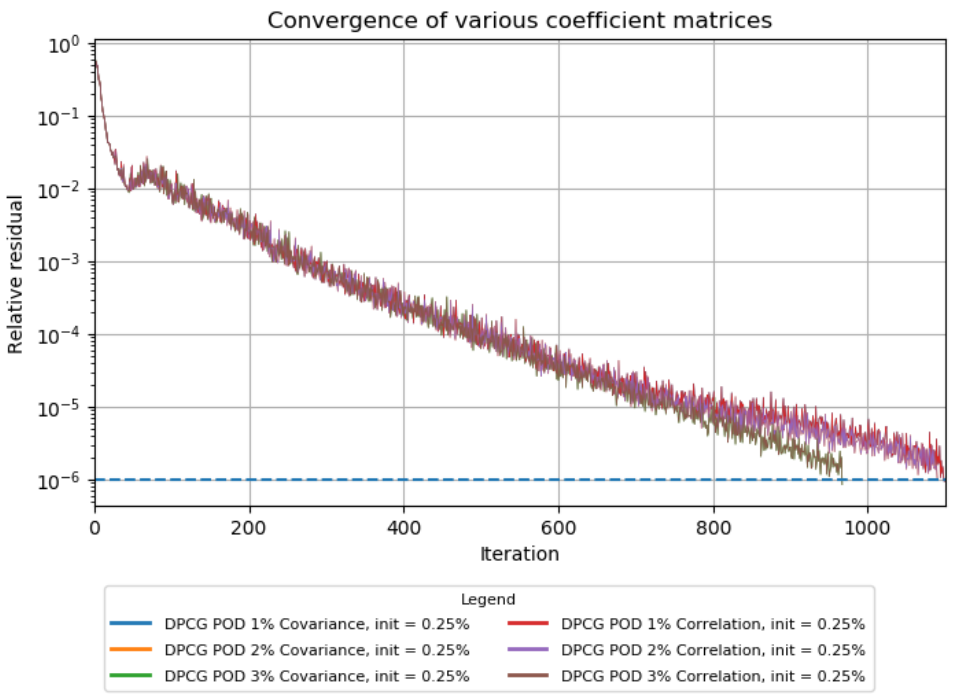


Figure 10.73: Comparison between covariance matrix and correlation matrix using the solution from the reduced original system with a reduction of 0.25% as the initial solution.

The eigenvalues of the data and covariance matrix rounded to 3 decimals:

Data	**Covariance**	**Correlation**
257698.407	185.069	0.715
174.828	29.357	0.227
29.279	9.306	$8.300 \cdot 10^{-2}$
9.298	2.739	$2.857 \cdot 10^{-2}$
2.739	2.078	$1.163 \cdot 10^{-2}$
2.077	0.892	$0.844 \cdot 10^{-2}$
0.892	0.660	$0.522 \cdot 10^{-2}$
0.660	0.586	$0.430 \cdot 10^{-2}$
0.586	0.507	$0.342 \cdot 10^{-2}$
0.506	0.462	$0.271 \cdot 10^{-2}$
0.460	0.363	$0.111 \cdot 10^{-2}$
0.362	$3.022 \cdot 10^{-15}$	$1.584 \cdot 10^{-17}$

Table 10.13: Eigenvalues of the data matrix on the left side and eigenvalues of covariance matrix on the right side.

Results of the POD + k-means algorithm applied on the covariance matrix:

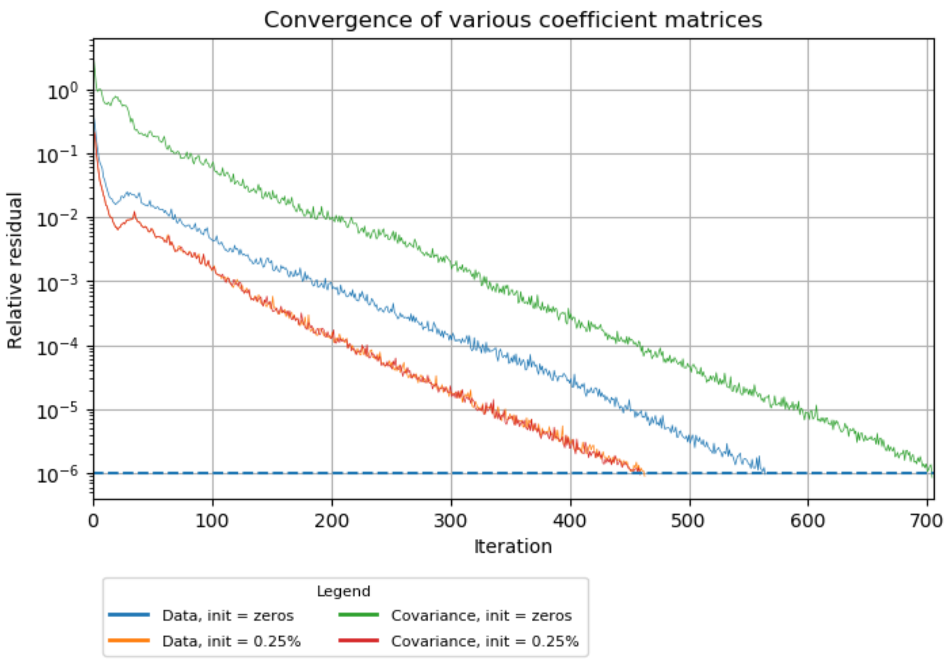


Figure 10.74: Comparison between POD + k-means+100 using the covariance matrix and subdomain size $l = 100$.

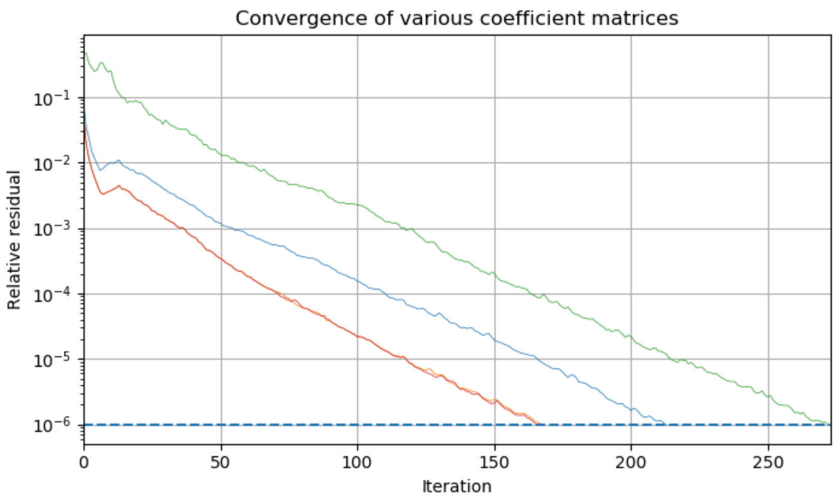


Figure 10.75: Comparison between POD + k-means++10 using the covariance matrix and subdomain size $l = 10$.

Method	θ_{min}	θ_{max}	$\frac{\theta_{max}}{\theta_{min}}$	iterations
Covariance $l = 100$	0.002577	48.39	18777.2	705
Covariance $l = 100$, init 0.25%	0.003431	48.39	14101.4	460
Covariance $l = 10$	0.002598	5.74	2208.9	273
Covariance $l = 10$, init 0.25%	0.003768	5.74	1522.95	167

Table 10.14: Ritz values of the POD + k-means method including only solutions from 0.25% and 3%.

Bibliography

[1] B. Alberts. Molecular biology of the cell, 2008.

[2] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[3] H. Bradford, Y. Masuda, P. VanRaden, A. Legarram, and I. Misztal. Modeling missing pedigree in single-step genomic blup. *Journal of Dairy Science*, 102(3), 2019.

[4] C. C. Chang, C. C. Chow, L. C. Tellier, S. Vattikuti, S. M. Purcell, and J. J. Lee. Second-generation plink: rising to the challenge of larger and richer datasets. *Gigascience*, 4(1):s13742–015, 2015.

[5] O. Christensen and M. Lund. Genomic prediction when some animals are not genotyped. *Genet Sel Evol*, 42(2), 2010.

[6] G. B. D. Cortes. *POD-Based Deflation Method For Reservoir Simulation.* PhD thesis, Delft University of Technology, Netherlands, 2019.

[7] G. Golub and C. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, 2013.

[8] C. Henderson, O. Kempthorne, S. Searle, and C. V. Krosigk. The estimation of environmental and genetic trends from records subject to culling. *Biometrics*, 15(2), 1959.

[9] C. R. Henderson. A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values. *Biometrics*, pages 69–83, 1976.

[10] Z. Liu, M. Goddard, F. Reinhardt, and R. Reents. A single-step genomic model with direct estimation of marker effects. *American Dairy Science Association*, 5833(5850), 2014.

[11] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[12] I. Misztal. Inexpensive computation of the inverse of the genomic relationship matrix in populations with small effective population size. *Genetics*, 202(2):401–409, 2016.

[13] R. A. Mrode. *Linear models for the prediction of animal breeding values.* Cabi, 2014.

[14] E. Mäntysaari and I. Strandén. Single-step genomic evaluation with many more genotyped animals. In *Proceedings of the 67th annual meeting of the european association for animal production (EEAP)*, Belfast, 29 Aug - 2 Sept 2016.

[15] S. C. Roth. What is genomic medicine? *Journal of the Medical Library Association: JMLA*, 107(3):442, 2019.

[16] Y. Saad. *Iterative Methods for Sparse Linear Systems.* Society for Industrial and Applied Mathematics, 2003.

[17] I. Strandén, K. Matilainen, G. Aamand, and E. Mäntysaari. Solving efficiently large single-step genomic best linear unbiased prediction models. *Journal of Animal Breeding and Genetics*, 134(3):264–274, 2017.

[18] J. Tang. *Two-Level Preconditioned Conjugate Gradient Methods with Applications to Bubbly Flow Problems.* PhD thesis, Delft University of Technology, 2008.

[19] A. van der Sluis. Condition, equilibration and pivoting in linear algebraic systems. *Numerische Mathematik*, 15, 1970.

[20] A. van der Sluis and H. van der Vorst. The rate of convergence of conjugate gradients. *Numerische Mathematik*, 48, 1986.

[21] H. van der Vorst and K. Dekker. Conjugate gradient type methods and preconditioning. *Journal of Computational and Applied Mathematics*, 24, 1988.

[22] J. Vandenplas, M. Calus, H. Eding, and C. Vuik. A second-level diagonal preconditioner for single-step snpblup. *Genetics Selection Evolution*, 51(30), 2019.

[23] J. Vandenplas, M. P. Calus, H. Eding, M. van Pelt, R. Bergsma, and C. Vuik. Convergence behavior of single-step gblup and snpblup for different termination criteria. *Genetics Selection Evolution*, 53(1):1–15, 2021.

[24] J. Vandenplas, H. Eding, M. Bosmans, and M. Calus. Computational strategies for the preconditioned conjugate gradient method applied to sssnpblup, with an application to a multivariate maternal model. *Genetics Selection Evolution*, 52(54), 2020.

[25] J. Vandenplas, H. Eding, M. Calus, and C. Vuik. Deflated preconditioned conjugate gradient method for solving single-step blup models efficiently. *Genetics Selection Evolution*, 50(51), 2018.

[26] C. Vuik and D. Lahaye. *Scientic Computing.* Delft Institute of Applied Mathematics, 2019.

[27] S. Waaijenborg and A. H. Zwinderman. Correlating multiple snps and multiple disease phenotypes: penalized non-linear canonical correlation analysis. *Bioinformatics*, 25(21):2764–2771, 2009.