UNIVERSITEIT GENT

FACULTY OF **BIOSCIENCE ENGINEERING**

Department of Mathematical Modelling, Statistics and Bioinformatics

High performance computing for large-scale genomic prediction

ir. Arne De Coninck

Thesis submitted in fulfillment of the requirements for the degree of Doctor (Ph.D.) of Applied Biological Sciences

Academic year 2015-2016

Supervisors:	Prof. dr. Bernard De Baets Department of Mathematical Modelling, Statistics and Bioinformatics Ghent University, Belgium
	Prof. dr. ir. Jan Fostier Department of Information Technology, Internet Based Communication Networks and Services Ghent University - iMinds, Belgium
Examination committee:	Prof. dr. ir. Wim Verbeke (Chairman)
	Prof. dr. Peter Dawyndt Prof. dr. Jan De Neve Prof. dr. ir. Geert Haesaert Dr. Andrés Legarra Dr. Steven Maenhout Prof. dr. Wim Vanroose
Dean:	Prof. dr. ir. Marc Van Meirvenne
Rector:	Prof. dr. Anne De Paepe

ir. Arne De Coninck

High performance computing for large-scale genomic prediction

Thesis submitted in fulfillment of the requirements for the degree of

Doctor (Ph.D) of Applied Biological Sciences

Academic year 2015-2016

Dutch translation of the title:
Hoogperformant computergebruik voor grootschalige genoomwijde voorspellingen

Please refer to this work as follows:
Arne De Coninck (**2016**). *High performance computing for large-scale genomic prediction*, PhD Thesis, Department of Mathematical Modelling, Statistics and Bioinformatics, Ghent University, Ghent, Belgium.

ISBN 978-90-5989-862-2

Acknowledgements - Dankwoord

Vier jaar zijn voorbijgevlogen sinds ik in november 2011 de keuze maakte om mijn horizon te verbreden door een doctoraatsstudie aan te vatten in een mij tot dan toe onbekend terrein. Gedurende die vier jaar stond ik er gelukkig niet alleen voor en heb ik de nodige steun en hulp gevonden bij enkele belangrijke personen.

Vooreerst dank ik mijn promotoren Bernard De Baets en Jan Fostier om mij deze unieke kans te geven om mezelf verder te ontplooien. Zij hebben er niet enkel voor gezorgd dat ik in alle rust en comfort kon werken aan dit doctoraatsonderzoek, maar ze hebben ook steeds geholpen in het uitzetten van duidelijke doelen en objectieven. Beiden hadden steeds een nuchtere en pragmatische kijk op mijn vorderingen en gaven op het juiste moment het nodige schouderklopje en de inspiratie om met volle moed het volgende probleem aan te pakken.

Uiteraard zijn ook collega's vitaal wanneer je vier jaar met een goed humeur wil gaan werken. Ook al waren de onderzoeksonderwerpen binnen de vakgroep vrij divers, toch kon ik bijna altijd met specifieke vragen wel bij iemand terecht, waarvoor dank. In het bijzonder wil ik toch mijn bureaugenoten Steffie, Jan, Pieter en Marlies bedanken, een leuke werkomgeving creëer je namelijk niet alleen.

Tot zover de personen die het dichtst bij het verwezenlijken van deze doctoraatsthesis stonden, maar daarbuiten hebben natuurlijk nog een aantal mensen een indirect aandeel in de succesvolle afronding van dit doctoraatsonderzoek. Niet in het minst verdienen mijn ouders, Patrick en Marleen, een bijzondere plaats in dit dankwoord voor de steun al sinds dag één van mijn bestaan. Het geeft een enorm goed gevoel te weten dat ik altijd op jullie kan rekenen. Ook mijn schoonfamilie heeft voor de nodige ontspanning en leuke momenten gezorgd, vooral op vrijdagavonden die steeds een perfecte afsluiter waren van de werkweek.

Er zijn ook heel wat vrienden waar ik veel aan te danken heb gehad tijdens de voorbije 4 jaar. Aan een opsomming ga ik mij niet wagen, ik ben bang dat ik anders iemand ga vergeten, maar als je dit leest, ben je het zeker waard vermeld te worden. Bedankt, beste vriend .....!

Als laatste houd ik nog de persoon aan wie ik het meest te danken heb. Domenique, jij bent niet enkel mijn vrouw en de moeder van mijn kinderen, maar vooral mijn grootste steun, mijn beste vriend, mijn raadgever, kortom mijn ideale supplement.

Arne De Coninck
Aaigem, 24/01/2016

"The road in front of us is long and it is wide
We've got beginner's luck, we've got it on our side
If you are willing, well, I think I'm qualified
And with beginner's luck we've gotta take the ride"

— Eels – Beginner's luck

Table of contents

Abbreviations

AI-REML Average Information Restricted Maximum Likelihood

ANOVA analysis of variance

API application programming interface

BLAS basic linear algebra subprograms

BLUE Best Linear Unbiased Estimation

BLUP Best Linear Unbiased Prediction

CPU central processing unit

CRS compressed row storage

DF derivative-free

DNA Deoxyribonucleic acid

EBV estimated breeding value

EM expectation-maximisation

GLSE generalized least squares estimator

GS Gauss-Seidel

GSA Genetics Society of America

G×E genotype-environment interaction

HPC high performance computing

LAPACK linear algebra package

MAS Marker-assisted selection

MKL Math Kernel Library

ML maximum likelihood

MME mixed model equations

MPI message passing interface

M×E marker-environment interaction

PBLAS parallel basic linear algebra subprograms

PCG preconditioned conjugate gradient

PPV positive predictive value

QTL quantitative trait locus/loci

Q×E QTL-environment interaction

RAM random access memory

REML Restricted Maximum Likelihood

RIL recombinant inbred line

RR-BLUP Ridge Regression Best Linear Unbiased Prediction

ScaLAPACK scalable linear algebra package

SNP single nucleotide polymorphism

TBV true breeding value

TPE target population of environments

TPR true positive rate

USDA United States Department of Agriculture

1 Introduction

"Like a drunk who looks for his lost keys under the lamp-post, because that is where the light is, scientists do certain experiments simply because they have the technologies available"

— Jane Calvert, 2012

1.1. General overview

In the 19th century Gregor Mendel discovered that certain properties of garden peas could be passed on to next generations, depending on certain invisible factors, which are now coined genes. The significance of his work wasn't recognized until the 20th century, but from that time genetics has played an important role in medicine, anthropology, psychology and, of course, plant and animal breeding. In the beginning of the exploitation of genetics in animal and plant breeding, it was still only a rather abstract concept and breeders tried to incorporate the genetic value or genetic merit of the individuals under study by modelling correlation between their phenotypic scores based on the pedigree of these individuals. Later on, the molecular side of genetics was studied intensively, leading to the knowledge that DNA is the molecule behind genetic inheritance and that DNA is composed out of a chain of nucleotides of which two strands are linked together in a helical structure. This knowledge was crucial for the development of DNA sequencing, which allows to read certain sequences of nucleotides from the DNA. Several of these sequences are now known to vary considerably in certain populations in only a single nucleotide and these variations are captured by a method called single-nucleotide polymorphism (SNP) genotyping.

The information from SNP genotyping experiments was introduced in animal and plant selection programs from the 1990s, when genetic merit was no longer only seen as a combined effect of all the genes of the individual, but it was modelled as the sum of this combined effect and the specific effect of certain SNPs. With the evolution of DNA sequencing, more SNPs became

available and some of these were categorized as having a significant effect on a certain quantitative trait. This particular locus on the DNA was then said to be a part of a quantitative trait locus (QTL). A locus is defined as the specific position of a gene or a short DNA sequence on the chromosome. These loci can occur in different forms known as alleles and a different allele may result in a difference in phenotype. In SNP genotyping it is mostly assumed that there are only two alleles possible at the specific locus and the difference is due to a difference in a single nucleotide. Nowadays, for dairy cattle over 36,000 different QTL are known to play a role in almost 500 different traits (`http://www.animalgenome.org/cgi-bin/QTLdb/BT/index`). This number is still rising, and it is thought that quantitative traits are not only controlled by QTL with a large contribution, but the combined effect of many QTL with only a minor contribution may play an important role in the further genetical improvement of the agronomical performance of animals and plants.

In order to take into account the combined effects of the many different QTL, dense SNP arrays were developed, sampling up to 700,000 SNPs across the whole genome and all this information is taken into account in so-called genomic selection programs. The estimation of the values of all these effects, referred to as genomic prediction, is almost impossible based on a low number of records, even with state-of-the-art methods. Therefore, a data-driven approach may represent a better way for optimising the predictability of the different effects that play a role in quantitative traits. Such a data-driven approach uses the abundance of information to deduce results without trying to optimise the analytical model as opposed to a theory-driven approach, where models are optimised to fit specific cases, usually based on smaller data sets. In fact, data-driven biology is emerging and technological advances in this area can lead to conceptual ones [1]. As the cost for genotyping is constantly decreasing due to technological advances, a first step for a data-driven genomic selection program, namely the genotypic data collection, is no longer an issue. However, the analysis of large-scale data sets still suffers from the computational burden associated with it. Although phenotypic data collection might still be an issue, historical records may be used and based on the analysis of these records, future phenotypic tests may be better planned when the predictions are more reliable.

A next step to take is thus enabling the analysis of these large-scale data sets to improve genomic selection programs. Nowadays, high performance

computing clusters are becoming more and more available not only in academia, but also in industry and even for private use. Efficiently applying this supercomputing power may accelerate advances in genomic prediction resulting in better estimates of genomic value of an individual, a better understanding of the origin of this genetic value and insights in how genetic value can change under different environmental conditions. For this reason, this dissertation tries to highlight the computational burden of genomic prediction and how this can be resolved by efficiently applying the computing power of a dedicated cluster. Moreover, two well-known and widely-used analysis methods in animal and plant breeding are implemented in such a way that the analysis of large-scale data sets on a high performance computing infrastructure becomes possible. The results show that indeed the data-driven research may dramatically increase the potential of genomic prediction and that high performance computing methods aid in the fast validation of the applied models. This dissertation thus tries to make new technology available for scientists to enable new ways of modelling or experimentation.

1.2. A road-map to this dissertation

This dissertation consists of an introductory part (part I), a part describing the main contributions (part II) and a conclusive part (part III). As such, it can be read in a linear way, because each chapter is the logical continuation of the previous one. Part I contains a very general but thorough introduction to genomic prediction as well as high performance computing, while part II describes two contributions of this dissertation in an animal and plant breeding context based on the methods that were introduced in the first part. Part III then summarises the results of part II and provides some future prospects based on the results of part II as well as current research practices.

1.2.1. Part I

Not only is it logical to start with an introductory part, it is also the perfect way for explaining the theoretical and historical foundations of the research fields at whose crossroads this dissertation is situated. A complete

introduction to both research fields would lead us too far, but the emphasis is on methods that will be used in part II for the applications in animal and plant breeding. A brief overview of the different chapters of this part is presented here to guide the reader on which topics might be of interest, before jumping into the contributions of this dissertation.

A very general introduction to genomic prediction is provided in Chapter 2, putting this dissertation in a historical perspective, while providing some insights of the computational aspects of genomic prediction. This might be of interest to all readers as some frequently used terms that are specific to genomic prediction are introduced. Also, it is always interesting to know more about historical aspects as Theodore Roosevelt once said: "The more you know about the past, the better prepared you are for the future".

Chapter 3 is dedicated to the use of linear mixed models in animal and plant breeding as this type of modelling still forms the cornerstone of current genomic prediction applications. The difference between linear regression and linear mixed models, who introduce random effects, is explained, together with the reason why these random effects are of importance in genomic prediction. The derivation of the best linear unbiased prediction (BLUP) for these random effects and the best linear unbiased estimation (BLUE) of the fixed effects in the linear mixed model is explained in depth. Furthermore, it is shown that the BLUP and BLUE can be found as the solution of a system of linear equations, better known as the mixed model equations. Finally, the computational aspects of solving these mixed model equations are emphasised, with common simplifications used in current genomic prediction settings to alleviate the computational burden.

The random effects in linear mixed models are assumed to follow a certain distribution, which is assumed to be the normal distribution throughout this dissertation. Mostly, the covariance structure of these random effects is known, but it still depends on some variance components. The estimation of these variance components is the subject of Chapter 4, again starting with a historical description of variance component estimation, but focusing on the method that is mainly used in genomic prediction, namely Restricted Maximum Likelihood. Subsequently, a convenient algorithm, the Average Information Algorithm, for finding the restricted maximum likelihood estimates is presented. Although the derivation of this algorithm is tedious, it provides for a computational convenience that is explained at the end of

Chapter 4, followed by a comparison of the computational burden of this algorithm to the computational complexity of other algorithms.

To resolve the computational demands of solving the mixed model equations and estimating the variance components for large-scale data sets, some high performance computing methods are presented in Chapter 5. At first, a little history of supercomputing is presented and the different ways of parallelisation as well as the limitations of parallel computing are explained. Next, distributed computing and sparse matrix algebra are discussed in depth, because these methods will be used in the following chapters of this dissertation. Finally, the application of high performance computing in genomic prediction is reviewed in a historical context and the opportunities for further developments in this area are highlighted.

1.2.2. Part II

The concepts explained in Part I are applied in both animal breeding and plant breeding, which is described in Part II. The first contribution of this dissertation is the application of distributed computing methods in an animal breeding context. This method was extended for being used in plant breeding where environmental conditions may play a role in the genetic ability of an individual. Therefore, sparse matrix algebra was coupled to the distributed computing methods that were used in the animal breeding context, because information coming from the different environments is usually only sparsely present.

Chapter 6 discusses the benefits of using distributed computing in an animal breeding setting, where the number of genotyped individuals becomes very large (up to 1 million) and the number of SNP markers used for genotyping can grow to 360,000. This implementation was called DAIRRy-BLUP, a parallel, distributed-memory ridge regression BLUP (RR-BLUP) implementation, that uses the Average Information algorithm for restricted maximum-likelihood estimation of the variance components. The goal of DAIRRy-BLUP is to enable the analysis of large-scale data sets to provide more accurate estimates of marker effects and breeding values. A distributed-memory framework is required since the dimensionality of the problem, determined by the number of SNP markers, can become too large to be analysed on a single computing node. The results showed that prediction accuracy is mainly influenced by increasing the number of genotyped

individuals included in the analysis instead of increasing the density of the SNP arrays used for genotyping.

In plant breeding, as explained in Chapter 7, phenotypic scores not only depend on the effect of genetic markers, but also on the environmental conditions. Moreover, the effect of markers may vary substantially under the influence of different environmental conditions and thus marker-by-environment interaction effects have to be taken into account. However, this may lead to a dramatic increase of the computational resources needed for analyzing large-scale trial data. A high performance computing solution, called Needles, is presented for handling such data sets. Needles is tailored to the particular properties of the underlying algebraic framework by exploiting a sparse matrix formalism where suited and by utilizing distributed computing methods to enable the use of a dedicated computing cluster. It is demonstrated that large-scale analyses can be performed within reasonable time frames with this framework. Moreover, by analyzing simulated trial data, it is shown that the effects of markers with a high environmental interaction can be predicted more accurately when more records per environment are available in the training data. The availability of such data and enabling their analysis with Needles may also lead to the discovery of highly contributing QTL in specific environmental conditions. Such a framework thus opens the path for plant breeders to select crops on these QTL, resulting in hybrid lines with optimized agronomic performance in specific environmental conditions.

1.2.3. Part III

The last part only consists of a single chapter, Chapter 8, which draws some general conclusions and highlights the future prospects of this research. The main conclusion from this dissertation is the fact that increasing the number of records in the analysis coming from genotyped individuals has a significant effect on the prediction accuracy of breeding values as well as on the predictability of the effects of the genetic markers. This increased predictability can lead to the detection of highly contributing QTL and QTL that have a variable effect in different environments. This provides a lot of potential for future research where crop or animals may be selected for specific types of environmental conditions and even for certain weather types. As weather predictions are always subject to uncertainty, risk minimisation

for yield loss when the weather is predicted incorrectly will play an important role in future genomic prediction.

1.A. Notational conventions

The use of mathematics to formalize and solve problems is a recurrent theme throughout this dissertation. Therefore, we will be needing a rather extensive mathematical notation. Even though several notational aspects are application specific, we will respect several conventions that are common in modern literature.

- **Vectors**

  Throughout this dissertation, we will often be using a vector notation. Vectors are denoted as boldface lowercase characters and are assumed to be column vectors. For example, $\mathbf{x}$ of dimension n represents an $n \times 1$ column vector. The transpose of the vector $\mathbf{x}$ is denoted $\mathbf{x}'$ and the ith element is denoted x_i. This means that we can write $\mathbf{x} = (x_1, \ldots, x_n)'$. Sometimes a vector will be indexed, for example, $\mathbf{u}_{\text{env}}$ and $\mathbf{u}_{\text{snp}}$ denote two different vectors of different dimension. The ith element of such vectors is then denoted $u_{\text{env},i}$ or $u_{\text{snp},i}$. Several "special" vectors are used: $\mathbf{0}_n$ denotes a $n \times 1$ vector of zeros; $\mathbf{1}_n$ denotes a $n \times 1$ vector of ones.

- **Matrices**

  Matrices are denoted as boldface capitalized characters. For example, $\mathbf{A}$ of dimension $m \times n$ denotes an $m \times n$ matrix of real numbers. Square $n \times n$ matrices are referred to as square matrices of dimension n. The ith row of a matrix $\mathbf{A}$ is denoted $\mathbf{A}_{i,\cdot}$; The jth column is denoted $\mathbf{A}_{\cdot,j}$. Moreover, the element in the ith row of the jth column is denoted $\mathbf{A}_{i,j}$ and will also be referred to as element (i, j) of $\mathbf{A}$. Sometimes, when notation allows it, the element in the ith row of the jth column of $\mathbf{A}$ is denoted $a_{i,j}$.

  $\mathbf{A}'$ denotes the matrix transpose of $\mathbf{A}$. For a square $n \times n$ matrix $\mathbf{A}$, $\mathbf{A}^{-1}$ denotes the inverse of $\mathbf{A}$ (assuming that its inverse exists). A matrix can also be subdivided into submatrices denoted as $\mathbf{A}_{(i,j)}$, for

example:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{(1,1)} & \mathbf{A}_{(1,2)} \\ \mathbf{A}_{(2,1)} & \mathbf{A}_{(2,2)} \end{bmatrix}.$$

The element (i, j) of $\mathbf{A}_{(1,2)}$ is then denoted $\mathbf{A}_{(1,2)_{i,j}}$.

The $n \times n$ identity matrix is denoted $\mathbf{I}_n$. A rectangular $m \times n$ matrix filled with zeros is denoted $\mathbf{0}_{m \times n}$. Sometimes, when the dimensions are not known or not specifically mentioned, $\mathbf{0}$ can denote a matrix filled with zeros of any dimension and $\mathbf{I}$ the identity matrix of an unspecified dimension.

The determinant of a matrix $\mathbf{A}$ is denoted $|\mathbf{A}|$. The trace of a matrix $\mathbf{A}$ is denoted $\text{tr}(\mathbf{A})$.

- **Functions**

  A generic function f with as input x is denoted $f(x)$, and for a vector input $\mathbf{x}$ this becomes $f(\mathbf{x})$. The partial derivative of a function with respect to the variable x_i is denoted $\frac{\partial f}{\partial x_i}$. Moreover, the gradient vector of f is denoted $\nabla f(x_i)$ and the Hessian matrix is denoted $\mathbf{H}$.

- **Random vectors**

  There is also a difference between fixed vectors, denoted by Greek symbols and random vectors, denoted by Latin characters. The difference between the two is explained more thoroughly in Chapter 3, but in short, random vectors are sampled from a probability distribution, while fixed vectors are not. In the model $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u}$, $\mathbf{y}$ and $\mathbf{u}$ are random vectors, while $\boldsymbol{\beta}$ is a fixed vector. Estimates and predictions of vectors that are used as effects in models are denoted by a hat, for example $\hat{\mathbf{u}}$ is the predicted value of the random vector $\mathbf{u}$.

  The expected value of a random vector $\mathbf{u}$ is denoted $\text{E}(\mathbf{u})$ and the covariance matrix of such a vector is denoted $\text{var}(\mathbf{u})$. The covariance matrix between two different random vectors $\mathbf{u}$ and $\mathbf{e}$ is denoted $\text{cov}(\mathbf{u}, \mathbf{e})$.

- **Probability distributions**

  The only probability distribution discussed in this dissertation is the normal distribution and a random vector $\mathbf{y}$ that is sampled from a

normal distribution with expected value $\boldsymbol{\beta}$ and covariance matrix $\mathbf{V}$ is denoted

$$\mathbf{y} \sim \mathcal{N}(\boldsymbol{\beta}, \mathbf{V}) \,.$$

PART I

HISTORY AND THEORETICAL BACKGROUND

2 Computational implications of genomic prediction

2.1. A short history of genomic prediction

The goal of genomic prediction is improving agronomic performance of plants or animals by making use of information about their genetic constellation. Sometimes it is also referred to as genomic selection, but where genomic selection puts the emphasis on the selection process, performed by the breeder, genomic prediction is actually a part of this process, focused on the analysis of the data and issuing recommendations to the breeder. Marker-assisted selection (MAS) is also used as a synonym for genomic prediction, however, the distinction between MAS and genomic prediction lies in the number of markers used in the predictive model. MAS usually only relies on a few markers, while we talk about genomic prediction when more than thousand markers across the whole genome are used. Although genetic effects were already discovered by Gregor Mendel in the 19th century, it wasn't until the beginning of the 20th century that genetics became a widely known concept in breeding. At first, studies were focused on qualitative traits, resulting in distinct discrete phenotypes. For instance, Mendel's garden peas had green versus yellow pods or round versus wrinkled seeds. These qualitative traits are mostly controlled by only one or a few genes and are in general not influenced by environmental conditions. Moreover, the gene has commonly only two alleles, a dominant one and a recessive one. In most multicellular organisms two copies of a gene are present, each inherited from one of the parents, and thus also two alleles are present. If these alleles are both the same, the organism is said to be homozygous with respect to that gene and if they are different, the organism is said to be heterozygous with respect to that gene. When the organism is homozygous for the recessive allele, the phenotype is different than when the organism is heterozygous or homozygous for the dominant allele.

Although qualitative traits might be useful in breeding, e.g. for disease resistance, nowadays the main interest for breeders lies in quantitative traits. As opposed to qualitative traits, quantitative traits don't result in

different categories of phenotypes, but display a continuum of phenotypes and are mostly heavily influenced by the environment. The most important quantitative trait for breeders is yield. For plants this is mostly expressed in tonnes/hectare, while for animals this can be kg/animal when breeding for meat or l/animal in dairy farming. It is also known that the genes that have an effect on the quantitative trait do not act independently and their combined effect is the product of additivity, dominance and epistasis. Additivity means that the total effect is relative to the number of times the alleles that influence the trait are present. If the total effect of two equal alleles at the same locus is smaller than simply doubling the effect of when the allele is present only once, dominance effects play a role. When the total effect of two alleles at two different loci is greater than the sum when only one of the alleles is present, it is said that epistasis is present. So dominance effects are due to the interaction of genes at the same locus, while epistasis is due to the interaction between genes at different loci. Because these effects are difficult to model, due to the fact that many interactions are possible, quantitative traits are mostly modelled by only considering the additive effects, as will also be the case further on in this dissertation. Another important concept in quantitative trait breeding is heritability, which is defined as the amount of genetic variance relative to the amount of phenotypic variance. This measure can be translated as the amount of variance in the phenotype that is due to genetic effects and thus a trait with low heritability is more influenced by non-genetic, mostly environmental effects. Such traits are thus harder to optimise by only selecting on the phenotypic performance, as this performance is a bad indicator of the genetic potential of the individual.

Quantitative trait breeding is nothing new, and has been performed for decades, but without the knowledge of the genetic constellation of a plant or animal. Crops with a high yield have always been selected above crops with a lower yield and farmers have also been aware that beneficial properties of plants can be passed on to their progeny. This type of phenotypic selection was commonly used to achieve a high-quality pure line. In animal breeding this led to intense inbreeding, not only due to difficulties in transportation, but mainly due to breeding prize males with their daughters and granddaughters in search of even better performing progeny. However, these inbred lines often grew sterile, which was only fully understood when Mendelian inheritance became a wide-known concept. To counter this decrease in

fertility, the inbred lines were crossed with other inbred lines, to develop a new crossbreed. Finally, this resulted in a constant cycle of inbreeding in search of high-quality lines and crossbreeding for avoiding degenerative properties. First improvements in plant breeding originate from the late 18th century when intensive and systematic investigations were performed with hybrid plants [2]. From these early studies, it was learned that in nature, self-pollination was generally avoided in order to take advantage of crossbreeding. This led to a quote from Charles Darwin, stating in his book Cross and Self Fertilization in the Vegetable Kingdom published in 1876: "the first and most important conclusion which may be drawn from the observations given in this volume, is that cross-fertilization is generally beneficial and self-fertilization injurious." Therefore, seeds for future generations came from open-pollinated cultivars, which resulted in a mixture of many different hybrids.

George Shull was one of the first scientists who made use of this knowledge to come up with enhanced breeding schemes for creating hybrid maize lines. He did this by using only pure inbred lines in separate fields, so these inbred lines could still be selected on their performance. But in one of his landmark papers, Shull also recognized that selfing towards homozygosity leads to inbreeding depression and that these inbred lines should be crossed to produce the best single crosses [3]. His findings led to the rapid introduction of hybrid maize in corn farms in the USA from the 1930s. At first double-cross maize hybrids were used, which are the progeny of the crossing of two single crosses, because they enabled the economic production of hybrid seed. However, double crosses are less uniform than single crosses, because a single cross is the result of crossing two pure lines that are homozygous in most alleles and so single crosses of the same pure lines will have almost the exact same genotypes. As double crosses are the result of crossing two single crosses that are likely to be heterozygous in some alleles, segregation occurs and so double crosses of the same single crosses can have varying genotypes. This of course makes it harder to predict the performance of these double crosses as they combine the properties of 4 inbred lines. Their introduction already increased grain yields significantly, but when single crosses became available in the 1960s due to the availability of new inbreds with higher seed yields, grain yield increased with an impressive rate of 0.129 t/ha per year. As a result, current maize hybrids have a yield that is 5 times higher than their prehybrid ancestors [4]. Nonetheless, food demands keep

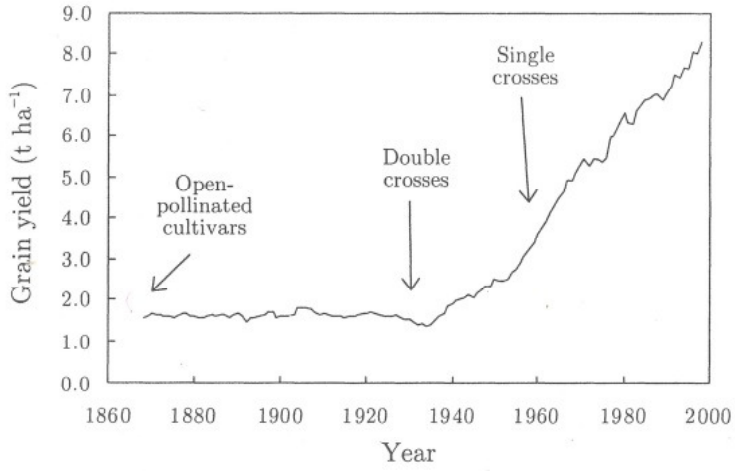

Figure 2.1: Five year moving average of the grain yield in the USA. Source [5]

on rising due to an ever increasing human population and thus even better performing hybrids are required. Therefore, plant breeders, but also animal breeders, make use of advanced statistical models and genetical information to constantly improve the production yield of their crop.

Statistics have always been of great help to animal and plant breeders, but while the latter focused on experimental design and analysis of trial data, the former were more interested in predicting breeding values for candidates of selection. The reason for this distinction in interest is obvious. First, reproduction of plants is faster and more numerous than reproduction of domestic animals, resulting in more and faster data acquisition. Secondly, testing of plants is a lot easier as inbred lines exist and as such these inbred lines and their single crosses can be used as checks in multiple environments to estimate the environmental impact on the agronomic performance. Therefore, plant breeders mainly relied on the outcome of multiple-environment trials to select the lines with a better performance. In animal breeding, data acquisition is a lot harder and it should not come as a surprise that especially to dairy cattle breeding, statistics has always made important contributions. The breeding of dairy cattle is confronted with the problem that performance can only be deduced directly for female cows, while the breeding value of male sires can only be estimated from the performance of their progeny. This is very time consuming, since breeders need to wait until the progeny is fully grown to estimate the breeding value of the sire. Furthermore, the data

is usually very unbalanced since only the bulls with the best progeny are used further on in the breeding process. Therefore, from the beginning of the 20th century, statistical models were applied to make use of information of relatives and to deal with the unbalancedness of the data.

Although it was originally conceived for studying human genetic variability, Fisher's infinitesimal model, stating that observations are a result of a large number of genetic additive factors and some residual, still is the statistical genetic point of departure for estimating breeding values in dairy cattle [6]. But the biggest contribution to the routine use of statistical models in animal breeding undoubtedly came from Henderson. He introduced linear mixed models in animal breeding, together with different methods to estimate variance components used in these models [7]. Most importantly, he extended greatly the use of matrix notation, leading to the still widely-used mixed model equations, which yield as solution the best linear unbiased predictions (BLUP) of the effects in a linear mixed model [8, 9]. Initially, genetic effects were modelled by assuming that each individual had a different genetic value and correlations between these individual effects were based on the pedigree structure. However, the rise of DNA sequencing starting from the 1970s, made it possible to also include information from genetic markers.

Genetic markers are identifiable variations in the genotype that can consist of a long or short DNA sequence and in animal breeding they were quickly used to investigate inheritance of quantitative traits [10]. At the end of the eighties, the linear mixed models as introduced by Henderson were transformed to enable the inclusion of information coming from these genetic markers [11, 12]. In practice, an effect was attributed to the genetic markers instead of attributing a single genetic effect to an individual. It marked the start of so-called marker-assisted selection, where information about the pedigree and of genetic markers were combined to estimate the agronomic potential of plants and mainly animals. Although it was predicted to yield 8 to 38 % of genetic gain, the usefulness of the sparse genetic markers is questionable when traits are controlled by a large number of loci and a population with a high genetic variability is under study [13].

The new millennium came with some new opportunities. Next-generation sequencing methods were implemented in commercial DNA sequencers, enabling genotyping across the whole genome with a dense marker map. The landmark paper by Meuwissen et al. from 2001 introduced the concept

of genomewide selection using information of genetic markers across the entire genome for the prediction of breeding values [14]. This paper clearly showed that using the BLUP estimation procedure with all marker effects included outperforms the least squares regression technique, which first has to preselect the informative markers. Moreover, the choice of a prior distribution for the genetic marker effects did not have a significant impact on the accuracy of the predicted breeding values. Therefore, it was a first hint that the normal assumption in the BLUP methodology, leading to the linear mixed model equations, is not a limiting factor. Especially because BLUP is far less computer intensive than other Bayesian methods, it can be considered as a viable method for estimating genetic marker effects.

Continuous improvements in genotyping led to the availability of more than 50,000 genetic markers for dairy cattle by 2008, implemented in a commercial DNA sequencer, the Illumina BovineSNP50 BeadChip [15]. These markers are single nucleotide polymorphisms (SNPs), which are defined as variations of the DNA at a single nucleotide that are common within a given population. They are very practical in use as they mostly only have two alleles, i.e. only two nucleotides are possible at the specific locus of the SNP. Therefore, SNPs are mostly presented as a binary code where 0 stands for the most frequent allele and 1 stands for the least frequent allele. Combining SNP codings for the maternal and paternal allele is performed by the summation of the binary code of the two chromosomes. As such 0 stands for homozygosity for the least frequent allele, 1 stands for heterozygosity and 2 stands for homozygosity for the most frequent allele. As the development of the BovineSNP50 assay was a collaboration with the US department of agriculture (USDA), the introduction of this large amount of SNP marker information in a BLUP estimation of the breeding values was also achieved by them [16]. This information was introduced by deriving relationships between individuals based on their SNP genotypes instead of using pedigree information as was already suggested in 1997 by Nejati-Javaremi et al. [17]. Although BLUP estimations resulted in fairly good estimates of the breeding values, several approaches were put forward, assuming a different distribution of the genetic marker effects [18, 19, 20]. For some data sets these more advanced methods resulted in a slightly higher prediction accuracy of the breeding values, but many studies also showed that the gain in performance was only very little compared to BLUP [21, 22, 23, 24]. In these studies and also further on in this thesis performance is quantified as the Pearson correlation between the

estimated values and the true values. This measure is commonly used in breeding applications as a linear correlation between true breeding values and estimated breeding values is considered important since breeders want to select the best performing individuals over worse performing individuals. Because a non-normal distribution of the marker effects has not shown to improve the performance of the predictive models, this dissertation will only focus on the BLUP methodology, applied to large-scale data sets.

Even though these methods mostly found their origin in animal breeding and more specifically in dairy cattle breeding, the plant breeding community has since the 1990s started to use the same methods to predict the performance of new hybrid breeds [25, 26]. But for plants there is an additional difficulty in the fact that the performance of plants is heavily influenced by environmental conditions. For animals this is of minor importance as environmental conditions are mostly under the control of the breeder, while plants are impacted by the climate, soil properties and fertilization. Not all plants behave equally under these varying environmental conditions, e.g. some families of plants will thrive better under sunny conditions, while others grow better under rainy conditions. Such effects are modelled by interactions between the genotype and the environment, so-called G×E effects [27]. The increased use of genetic markers for genotyping plants has caused the adaptation of the models used in animal breeding by introducing marker-by-environment interaction effects [28, 29, 30, 31]. This can lead to a huge increase in computational resources needed for analysis of the data and due to the never-ending growth of such data, the computational challenge for analysing such data is a hot topic.

2.2. Computational aspects

From the beginning of the use of linear mixed models and the BLUP estimation procedure, some computational aspects have been a challenge and/or limiting factor. Henderson already proposed 3 methods for estimating variance components in linear mixed models [7]. The method with the most satisfactory results had as a downside that it was not always computationally feasible and the other two were examples of balancing between simplicity and accuracy. Computational aspects were also one of the reasons why Henderson transformed the BLUP estimation into solving a matrix equation.

The BLUP estimation procedure involves the inversion of the covariance matrix of the observations, which can become a costly and difficult operation. This was certainly the case in times when a personal computer was not yet available [9].

With the rise of the microprocessor in the 1970s and 1980s, applicability of the BLUP procedure increased to ever growing data sets. However, the collection of data was still faster than Moore's law [32], and thus specialized algorithms had to be applied for enabling the analysis of the data. As Henderson already had problems calculating the variance components, this still was the bottleneck for the BLUP methodology. There were different algorithms available for estimating the variance components, but they all had in common that they required the inversion of the coefficient matrix of the mixed model equations. As these coefficient matrices were mostly sparse, i.e. contained a lot of zeros, efficiency of the computer algorithms could be optimized by exploiting this sparsity [33, 34]. Even though a "supercomputer" was already used by Misztal, very large problems still took too much time to be evaluated [34].

The application of genetic markers in genomic prediction led to new computational problems. The replacement of a single genetic effect attributed to the genetic merit of an individual by a summation of the genetic marker effects resulted in a densely filled coefficient matrix of the mixed model equations, restricting the use of a sparse matrix formalism. Meuwissen et al. already stumbled upon this problem by the fact that the coefficient matrix did no longer fit in the RAM memory of his Pentium500 PC and he needed to use an iterative technique, which increased the computer time quadratically with the number of effects to be estimated [14]. Luckily, computing power increased dramatically in the following years, but the number of genetic markers also increased with gigantic leaps. It made Legarra and Misztal think about different options to overcome the demand for considerable computing resources in time as well as in storage and they proposed several strategies to efficiently solve the mixed model equations [35]. As a conclusion, they proposed to use iterative techniques such as preconditioned conjugate gradient (PCG), which do not require the set-up of the mixed model equations, unless when the number of observations should increase dramatically. The PCG method's complexity is in fact proportional to the number of observations and therefore, a direct solution using Cholesky decomposition becomes a viable alternative, since this method's complexity only depends

on the number of effects included in the linear mixed model.

Due to the availability of even larger SNP arrays, VanRaden showed that an equivalent model exists where again individual genetic effects are modelled for which a relationship matrix is deduced from the genetic marker information [16]. The downside of this method, was the need for inversion of this genomic relationship matrix, which could be unfeasible when this matrix is singular. Such a framework paved the way for a method where genomic selection could be improved by combining the information of the pedigree, genetic markers and phenotypic information of non-genotyped individuals [36]. Nonetheless, it was already acknowledged that this algorithm was limited in the number of genotyped individuals to be included. As such most studies concentrated on data sets with low numbers of genotyped individuals ($< 10,000$) [37, 38]. Advances in the field of genomic prediction were also oriented towards larger SNP assays, while the number of genotyped individuals included in the analyses did not increase significantly [39, 40].

However, it had already been shown theoretically and experimentally that for a major improvement in prediction accuracy of the breeding values, the number of genotyped individuals included in the analysis should increase dramatically, especially when the trait under study has a low heritability [21, 41]. The challenges to be met with such large-scale data sets were pointed out by Cole, stating that there is a need to be able to process and analyze large-scale data sets using high-performance computing techniques [42].

The research in this dissertation is situated in that specific field. Namely, applying high performance computing techniques to enable the analysis of large-scale data sets for improving the accuracy of the prediction of breeding values of animals in plants in the context of genomic selection.

3 Linear mixed models in breeding applications

3.1. Introduction

In this chapter, a summary of the properties of a linear mixed model is given, together with an extensive derivation of the methods used further on for estimating the fixed and random effects. The difference between these two types of effects that make up a linear mixed model are explained further on. For a more thorough description of linear mixed models, their history and different estimation procedures, the book "Variance Components" by Shayle R. Searle is highly recommended [43]. Not only was Searle an expert in linear and mixed models, he was also a PhD student of Charles Henderson, making him familiar with animal breeding and as such many of the examples in his book come from animal breeding experiments.

A linear mixed model is an extension of a linear model where the n observations $\mathbf{y}$ are a linear function of k fixed effects making up the vector $\boldsymbol{\beta}$ and a vector $\mathbf{e}$ of n residuals:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{e}\,,$$

where $\mathbf{X}$ is the incidence matrix for the effects, which is in this thesis always fixed and contains no random variables, and thus $\mathrm{var}(\mathbf{y}) = \mathrm{var}(\mathbf{e}) = \mathbf{V}$. The residual $\mathbf{e}$ is defined as $\mathbf{y} - \mathrm{E}(\mathbf{y})$ and as such it holds that $\mathrm{E}(\mathbf{e}) = \mathbf{0}_n$. The Generalized Least Squares Estimator (GLSE) for the effects, which is unbiased, consistent and efficient, is:

$$\mathrm{GLSE}(\boldsymbol{\beta}) = \left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}\,.$$

This estimator is not always useful in practice, because the covariance matrix $\mathbf{V}$ is mostly not known, and thus Ordinary Least Squares (OLS), ignoring the covariance matrix, is still commonly used, which is also computationally faster. Nonetheless, iterative solutions for obtaining the GLSE exist when $\mathbf{V}$ is not known, but when $\mathbf{V}$ is incorrectly estimated, these methods may result in an estimator that is no longer efficient. We will not go into details about comparing these methods as we mainly introduce GLSE here as an

introduction to linear mixed models, where the GLSE for the fixed effects can be obtained in a more elegant way.

Fixed effects models have already proven their value in many different areas, for instance, in clinical trials where k different treatments for reducing blood pressure are compared against each other. The difference in blood pressure for each patient is the observation vector $\mathbf{y}$, while the vector $\boldsymbol{\beta}$ consists of the effects of the k different treatments. However, this model can be extended when the clinical trial is performed over a number of randomly chosen clinics, but the different treatments are of course to be used globally and not be limited to the clinics where the treatments were tested. The effects of the clinics are then modelled as random effects, since we only possess of data coming from a random sample of the much larger population of clinics where the treatments should be applied. This is, therefore, a good example of a linear mixed model, where fixed as well as random effects are modelled. A general form of this kind of models is:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}\,,$$

where $\mathbf{u}$ is the vector of l random effects and $\mathbf{Z}$ is the incidence matrix for these random effects. In its most general form, the random effects may be considered as drawn from any kind of distribution with different possibilities for the covariance structure, but commonly the random effects have zero mean and a covariance structure that depends on only a few variance components. This can be summarized as:

$$\mathrm{E}(\mathbf{u}) = \mathbf{0}_l \qquad \mathrm{var}(\mathbf{u}) = \mathbf{G}(\boldsymbol{\phi})\,,$$

where E represents expectation, $\boldsymbol{\phi}$ is the vector of variance components on which the covariance structure relies and $\mathbf{0}_l$ is the null vector of length l. Moreover, a universally adopted assumption is the fact that the residuals and the random effects are not correlated and statistically independent and thus

$$\mathrm{cov}(\mathbf{u}, \mathbf{e}) = \mathbf{0}_{l\times n}\,,$$

with $\mathbf{0}_{l\times n}$ a matrix filled with zeros and as dimensions the number of random effects l and the number of observations n. As such, the entire covariance structure for the observations, with $\mathrm{var}(\mathbf{e}) = \mathbf{R}(\boldsymbol{\gamma})$ and $\boldsymbol{\gamma}$ a vector of variance

components on which the variance of $\mathbf{e}$ relies, has the form:

$$\text{var}(\mathbf{y}) = \mathbf{Z}\mathbf{G}(\boldsymbol{\phi})\mathbf{Z}' + \mathbf{R}(\boldsymbol{\gamma})\,.$$

In the clinical trial example, we modelled the clinics as random effects, because the treatments were only tested in a random sample of clinics, and the results needed to be inferred towards a larger population of clinics. Nonetheless, if we were interested in the specific effect of the clinics where the treatments were tested and the treatments would only be applied in those clinics, it would be possible to model them as fixed effects. It is thus not always very clear when to model an effect as random or fixed. There are two questions that could be posed to discriminate between the two:

- Are the different effects a sample from a larger population and do we want to extend the results of the analysis towards the entire population?
- Do we possess of prior information, indicating that the realised values of the effects come from a probability distribution?

If the answer is yes to one of these two questions, it might be wise to model the effects as random effects instead of fixed effects. Sometimes, as in the case of the clinical trial, it is much simpler to make the distinction, as we are mostly only interested in the effect of the treatment and not specifically in the effect of a certain clinic. The clinics are, therefore, modelled as random effects because it enables us to distinguish variance due to clinical effects from the residual variance due to the difference in reaction to a treatment by the different patients. However, apart from being interested in the variation due to the random effects, it may be required to possess of an estimation of the realised value for the random effect. Suppose that the random sample of clinics should also be evaluated on their performance, for instance to determine where procedures were not followed in accordance with best practices, then, a ranking of the different clinics is required. Note that such an estimation is actually called a prediction of the random effect, while fixed effects are estimated. It should now be clear that the distinction between fixed or random effects does not depend on whether we want to know the realised values of that effect, but is determined by the assumption that the realised values of that effect are sampled randomly from a probability distribution. The prediction of these random effects can be done in a couple of ways, but we will focus here on the Best Linear Unbiased Prediction

(BLUP) as will be discussed in Section 3.2.

In dairy cattle breeding, linear mixed models where first used to analyse the milk production records of a herd of cows. As breeders were mainly interested in the breeding value of a sire, the milk production records of the progeny of a sire were analysed as having an effect of the sire and of the herd [9]. Both can be considered as random effects, because they can be treated as random samples of a larger population. At first the covariance structure was kept very simple by modelling the covariance matrix as a constant diagonal matrix. However, Henderson quickly saw hail in introducing information about the relationships between the animals as a correlation between their genetic effects [44]. A commonly used model can then be written as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}_1\mathbf{s} + \mathbf{Z}_2\mathbf{a} + \mathbf{e}\,, \tag{3.1}$$

with

$$\begin{aligned} \mathrm{var}(\mathbf{s}) &= \sigma_s^2\mathbf{S}\,, \\ \mathrm{var}(\mathbf{a}) &= \sigma_a^2\mathbf{A}\,, \\ \mathrm{var}(\mathbf{e}) &= \sigma_e^2\mathbf{I}\,. \end{aligned}$$

$\mathbf{a}$ is a vector with random effects representing genetic merit of an animal, $\mathbf{s}$ is a vector representing other random effects (e.g. herd effects), $\mathbf{S}$ is known and nonsingular and $\mathbf{A}$ represents the relationship matrix. This relationship matrix is symmetric, with diagonal elements $1 + f_i$, where f_i is the inbreeding coefficient of animal i and off-diagonal elements depend on the relationship between the different animals, derived from the pedigree, and their inbreeding coefficients [45].

Table 3.1: Example of a population structure

Individual	Parents	Herd
1	unknown	1
2	unknown	1
3	1 and unknown	2
4	1 and 2	2
5	3 and 4	3
6	1 and 4	3

As an example, let's look at a population with a structure as defined in Table 3.1. A simple way of calculating the numerator relationship matrix

A is using the recursive formula stating that the coefficient of correlation between two individuals is half of the sum of the correlation coefficients of the parents of one of the individuals with the other individual. Moreover, the inbreeding coefficient is half of the coefficient of correlation between the two parents of the individual. In this way, the elements of the numerator relationship matrix **A** can be calculated from the upper left corner to the lower right corner:

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0.5 & 0.5 & 0.5 & 0.75 \\ 0 & 1 & 0 & 0.5 & 0.25 & 0.25 \\ 0.5 & 0 & 1 & 0.25 & 0.625 & 0.375 \\ 0.5 & 0.5 & 0.25 & 1 & 0.625 & 0.75 \\ 0.5 & 0.25 & 0.625 & 0.625 & 1.125 & 0.5625 \\ 0.75 & 0.25 & 0.375 & 0.75 & 0.5625 & 1.25 \end{bmatrix}.$$

As there is only one record available per individual, $\mathbf{Z}_2$ is equal to the identity matrix of dimension 6. The covariance structure of the herd effects is usually assumed to be diagonal and as such **S** is the identity matrix of dimension 3 (= number of herds). The incidence matrix $\mathbf{Z}_1$ is then defined as

$$\mathbf{Z}_1 = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

The total covariance matrix **V** of the observations **y** will then have the following form:

$$\mathbf{V} = \sigma_s^2\mathbf{Z}_1\mathbf{Z}_1' + \sigma_a^2\mathbf{A} + \sigma_e^2\mathbf{I}_6$$

$$= \begin{bmatrix} \sigma_s^2+\sigma_a^2+\sigma_e^2 & \sigma_s^2 & 0.5\sigma_a^2 & 0.5\sigma_a^2 & 0.5\sigma_a^2 & 0.75\sigma_a^2 \\ \sigma_s^2 & \sigma_s^2+\sigma_a^2+\sigma_e^2 & 0 & 0.5\sigma_a^2 & 0.25\sigma_a^2 & 0.25\sigma_a^2 \\ 0.5\sigma_a^2 & 0 & \sigma_s^2+\sigma_a^2+\sigma_e^2 & \sigma_s^2+0.25\sigma_a^2 & 0.625\sigma_a^2 & 0.375\sigma_a^2 \\ 0.5\sigma_a^2 & 0.5\sigma_a^2 & \sigma_s^2+0.25\sigma_a^2 & \sigma_s^2+\sigma_a^2+\sigma_e^2 & 0.625\sigma_a^2 & 0.75\sigma_a^2 \\ 0.5\sigma_a^2 & 0.25\sigma_a^2 & 0.625\sigma_a^2 & 0.625\sigma_a^2 & \sigma_s^2+1.125\sigma_a^2+\sigma_e^2 & \sigma_s^2+0.5625\sigma_a^2 \\ 0.75\sigma_a^2 & 0.25\sigma_a^2 & 0.375\sigma_a^2 & 0.75\sigma_a^2 & \sigma_s^2+0.5625\sigma_a^2 & \sigma_s^2+1.25\sigma_a^2+\sigma_e^2 \end{bmatrix}.$$

With the advent of commercial DNA sequencing, this kind of model had to change to include information of genetic markers. Meuwissen et al. thus transformed the linear mixed model in Eq. (3.1) by replacing the random effect of genetic merit by adding the different effects of the genetic markers [14]. This resulted in the following model, omitting other random effects:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}\,,$$

with $\mathbf{u}$ the vector of genetic marker effects and $\mathbf{Z}$ their incidence matrix. In this way the relationship matrix was no longer of use, because relationships between individuals were naturally incorporated due to the fact that more related individuals shared more of the same genetic markers.

Until now it was left in the middle what kind of distributions were assumed for the random effects. Historically, most random effects were assumed normally distributed, with a covariance structure most suited to the setting where the model was used. Meuwissen et al. considered a few options for this covariance structure, assuming no correlations between the genetic marker effects. One of the options is a homoscedastic variance for all marker effects, leading to $\text{var}(\mathbf{u}) = \sigma_u^2\mathbf{I}$. Another option was a varying variance for each marker effect, sampled from a scaled inverted chi-square distribution, with the possibility that certain marker effects have zero variance and thus do not contribute to the trait. Such assumptions are equivalent to assuming that the part of the markers that contribute to the trait are sampled from a student t-distribution. These, and other so-called bayesian methods were compared against the normality assumption with little or no gain in prediction accuracy for the breeding values [19, 46]. Also, as will be shown later on, the assumption of normality leads to a predictor of the random effects $\mathbf{u}$ that is linear in $\mathbf{y}$ and such prediction methods lend themselves better to be scaled up with the use of high performance computing techniques. Therefore, from now on, random effects will be assumed normally distributed, with a covariance structure only depending on a few variance components.

The remainder of this chapter will be organised as follows:

- Section 3.2 will derive the best linear unbiased predictor for the random effects in a linear mixed model, under the assumption of normality of the random effects.
- Section 3.3 will give a derivation of the Mixed Model equations and

prove that the solution of these equations yields the best linear unbiased estimators/predictors for the fixed and random effects.

- Section 3.4 will review the computational aspects of solving these mixed model equations in a breeding perspective and will comment on some methods that have already been applied.

3.2. Best Linear Unbiased Prediction (BLUP)

The Best Linear Unbiased Predictors (BLUPs) for the random effects in a linear mixed model have first been found by Henderson in 1963, but were not called this way until 1973 [8, 47]. The prediction methodology of the BLUP for the random effects in a linear mixed model actually involves the Best Linear Unbiased Estimation (BLUE) of the fixed effects. Strictly speaking, the distinction between the two is that fixed effects are estimated, while an estimate of a realised value of a random effect is called a prediction. Nonetheless, BLUP is globally assumed as incorporating BLUE, because a BLUP of random effects requires BLUE of the fixed effects. The derivation presented here is slightly different than the original derivation of Henderson, as he did not explicitly use the fact that the random effects are normally distributed, but rather used Lagrangian multipliers to minimise the mean squared error of prediction. Note that the ending P of BLUP can stand for prediction as well as predictor, just as the ending E in BLUE can mean estimate, estimator or estimation depending on the context.

Our derivation starts with the following assumptions:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}\,,$$

with

$$\begin{aligned} \begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} &\sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{G} & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix}\right), \\ \mathbf{y} &\sim \mathcal{N}\left(\mathbf{X}\boldsymbol{\beta}, \mathbf{V}\right), \\ \mathbf{V} &= \mathbf{Z}\mathbf{G}\mathbf{Z}' + \mathbf{R}\,, \end{aligned} \tag{3.2}$$

where we have left out the dependence of the covariance matrices on the variance component vectors $\boldsymbol{\phi}$ and $\boldsymbol{\gamma}$, since in this chapter we will assume that these are known and $\mathbf{G}$, $\mathbf{R}$ and $\mathbf{V}$ are constant. The best estimator for

estimating a parameter usually means minimising the variance, but in this case the best predictor for $\mathbf{u}$ is defined as the predictor that minimises the mean squared error of prediction. The mean squared error for the prediction $\tilde{u}$ of a single random variable u is:

$$\mathrm{E}[(u-\tilde{u})^2] = \iint (\tilde{u}-u)^2 f(u,\mathbf{y})\mathrm{d}\mathbf{y}\mathrm{d}u\,,$$

where $f(u,\mathbf{y})$ is the joint density function of the random variables U and $\mathbf{Y}$ at the point $(u,\mathbf{y})$. For a vector of random variables $\mathbf{u}$, the above can be extended to:

$$\mathrm{E}[(\tilde{\mathbf{u}}-\mathbf{u})'\mathbf{A}(\tilde{\mathbf{u}}-\mathbf{u})] = \iint (\tilde{\mathbf{u}}-\boldsymbol{v})'\mathbf{A}(\tilde{\mathbf{u}}-\boldsymbol{v}) f(\boldsymbol{v},\mathbf{y})\mathrm{d}\mathbf{y}\mathrm{d}\boldsymbol{v}\,, \tag{3.3}$$

where $\mathbf{A}$ is any positive definite symmetric matrix. The best predictor which minimises Eq. (3.3) is known to be the conditional expectation of $\mathbf{u}$ given $\mathbf{y}$:

$$\text{best predictor} : \tilde{\mathbf{u}} = \mathrm{E}(\mathbf{u}|\mathbf{y})\,,$$

This is verified as follows, by denoting $\mathbf{u}_0 = \mathrm{E}(\mathbf{u}|\mathbf{y})$:

$$\begin{aligned}\mathrm{E}[(\tilde{\mathbf{u}}-\mathbf{u})'\mathbf{A}(\tilde{\mathbf{u}}-\mathbf{u})] &= \mathrm{E}[(\tilde{\mathbf{u}}-\mathbf{u}_0+\mathbf{u}_0-\mathbf{u})'\mathbf{A}(\tilde{\mathbf{u}}-\mathbf{u}_0+\mathbf{u}_0-\mathbf{u})]\\ &= \mathrm{E}[(\tilde{\mathbf{u}}-\mathbf{u}_0)'\mathbf{A}(\tilde{\mathbf{u}}-\mathbf{u}_0)] + 2\mathrm{E}[(\tilde{\mathbf{u}}-\mathbf{u}_0)'\mathbf{A}(\mathbf{u}_0-\mathbf{u})]\\ &\quad + \mathrm{E}[(\mathbf{u}_0-\mathbf{u})'\mathbf{A}(\mathbf{u}_0-\mathbf{u})]\,.\end{aligned}$$

If we look a bit closer to the second term, it can be seen that:

$$\begin{aligned}\mathrm{E}[(\tilde{\mathbf{u}}-\mathbf{u}_0)'\mathbf{A}(\mathbf{u}_0-\mathbf{u})] &= \mathrm{E}_y\left[\mathrm{E}_u[(\tilde{\mathbf{u}}-\mathbf{u}_0)'\mathbf{A}(\mathbf{u}_0-\mathbf{u})|\mathbf{y}]\right]\\ &= \mathrm{E}_y[(\tilde{\mathbf{u}}-\mathbf{u}_0)'\mathbf{A}(\mathbf{u}_0-\mathbf{u}_0)] = \mathbf{0}\,,\end{aligned}$$

because only $\mathbf{u}$ is not fixed given $\mathbf{y}$ and $\mathrm{E}_u(\mathbf{u}|\mathbf{y}) = \mathbf{u}_0$. As such, the mean squared error is composed of 2 terms, but we want to minimise it with respect to $\tilde{\mathbf{u}}$ and the last term does not depend on $\tilde{\mathbf{u}}$. Because $\mathbf{A}$ is a positive definite matrix, $(\tilde{\mathbf{u}}-\mathbf{u}_0)'\mathbf{A}(\tilde{\mathbf{u}}-\mathbf{u}_0)$ is always positive except when $(\tilde{\mathbf{u}}-\mathbf{u}_0)$ is the null vector $\mathbf{0}_l$. Therefore, the mean squared error is minimised by choosing $\tilde{\mathbf{u}} = \mathbf{u}_0 = \mathrm{E}(\mathbf{u}|\mathbf{y})$.

Given the assumptions in Eq. (3.2), an expression for $\mathrm{E}(\mathbf{u}|\mathbf{y})$ will be derived based on the conditional probability density function $f(\mathbf{u}|\mathbf{y})$. Bayes theorem

states that the conditional density of $\mathbf{u}$ given $\mathbf{y}$ can be found as:

$$f(\mathbf{u}|\mathbf{y}) = \frac{f(\mathbf{u},\mathbf{y})}{f(\mathbf{y})}\,. \tag{3.4}$$

The marginal density function of $\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \mathbf{V})$ is written more explicitly as:

$$f(\mathbf{y}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)}{(2\pi)^{n/2}|\mathbf{V}|^{1/2}}\,,$$

with n the number of elements in vector $\mathbf{y}$. The joint probability density function of $\mathbf{u}$ and $\mathbf{y}$ is defined as:

$$\begin{bmatrix}\mathbf{u}\\ \mathbf{y}\end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mathbf{0}\\ \mathbf{X}\boldsymbol{\beta}\end{bmatrix}, \begin{bmatrix}\mathbf{G} & \mathbf{G}\mathbf{Z}'\\ \mathbf{Z}\mathbf{G} & \mathbf{V}\end{bmatrix}\right)\,,$$

with $\mathbf{V}$ as defined in (3.2) and the exact form of the joint probability density function of $\mathbf{u}$ and $\mathbf{y}$ can be written as:

$$f(\mathbf{u},\mathbf{y}) = \frac{\exp\left(-\frac{1}{2}\begin{bmatrix}\mathbf{u}' & (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\end{bmatrix}\boldsymbol{\Sigma}^{-1}\begin{bmatrix}\mathbf{u}\\ (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\end{bmatrix}\right)}{(2\pi)^{(n+l)/2}|\boldsymbol{\Sigma}|^{1/2}}\,, \tag{3.5}$$

where

$$\boldsymbol{\Sigma} = \begin{bmatrix}\mathbf{G} & \mathbf{G}\mathbf{Z}'\\ \mathbf{Z}\mathbf{G} & \mathbf{V}\end{bmatrix}$$

and l is equal to the number of elements in vector $\mathbf{u}$. The determinant of such a block matrix, assuming $\mathbf{V}$ is invertible, can be found as:

$$|\boldsymbol{\Sigma}| = |\mathbf{V}|\left|\mathbf{G}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{Z}\mathbf{G}\right|\,, \tag{3.6}$$

while the inversion of $\boldsymbol{\Sigma}$ in a blockwise fashion is:

$$\begin{aligned}\boldsymbol{\Sigma}^{-1} &= \begin{bmatrix}\mathbf{G} & \mathbf{G}\mathbf{Z}'\\ \mathbf{Z}\mathbf{G} & \mathbf{V}\end{bmatrix}^{-1}\\ &= \begin{bmatrix}\mathbf{W} & -\mathbf{W}\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\\ -\mathbf{V}^{-1}\mathbf{Z}\mathbf{G}\mathbf{W} & \mathbf{V}^{-1}+\mathbf{V}^{-1}\mathbf{Z}\mathbf{G}\mathbf{W}\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\end{bmatrix}\,,\end{aligned} \tag{3.7}$$

with

$$\mathbf{W} = (\mathbf{G}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{Z}\mathbf{G})^{-1}\,.$$

For the division of $f(\mathbf{u},\mathbf{y})$ by $f(\mathbf{y})$, let us first look at the denominators of both probability density functions. Using the result of Eq. (3.6), the denominator of $f(\mathbf{u}|\mathbf{y})$ is:

$$\frac{(2\pi)^{(n+l)/2}|\boldsymbol{\Sigma}|^{1/2}}{(2\pi)^{n/2}|\mathbf{V}|^{1/2}} = (2\pi)^{l/2}|\mathbf{W}|^{1/2}\,.$$

The numerator is a quotient of two exponential functions which amounts to the subtraction of the arguments of these exponential functions:

$$\begin{aligned}
\log\left(\frac{f(\mathbf{u}|\mathbf{y})}{(2\pi)^{-l/2}|\mathbf{W}|^{-1/2}}\right) &= -\frac{1}{2}\begin{bmatrix}\mathbf{u}\\ \mathbf{y}-\mathbf{X}\boldsymbol{\beta}\end{bmatrix}'\boldsymbol{\Sigma}^{-1}\begin{bmatrix}\mathbf{u}\\ \mathbf{y}-\mathbf{X}\boldsymbol{\beta}\end{bmatrix}\\
&\quad -\frac{1}{2}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\\
&= -\frac{1}{2}\left(\mathbf{u}'\mathbf{W}\mathbf{u}-(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}\mathbf{Z}\mathbf{G}\mathbf{W}\mathbf{u}\right.\\
&\quad -\mathbf{u}'\mathbf{W}\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})+(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\\
&\quad +(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}\mathbf{Z}\mathbf{G}\mathbf{W}\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\\
&\quad \left.-(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)\\
&= -\frac{1}{2}\left(\left(\mathbf{u}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)'\mathbf{W}\mathbf{u}\right.\\
&\quad \left.-\left(\mathbf{u}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)'\mathbf{W}\mathbf{Z}\mathbf{G}\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)\\
&= -\frac{1}{2}\left(\mathbf{u}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)'\mathbf{W}\left(\mathbf{u}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)\,.
\end{aligned}$$

In the second step we made use of the explicit formula for the inverse of $\boldsymbol{\Sigma}$ as shown in Eq. (3.7). The conditional probability density function of $\mathbf{u}$ given $\mathbf{y}$ is thus:

$$f(\mathbf{u}|\mathbf{y}) = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{u}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)'\mathbf{W}\left(\mathbf{u}-\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\right)\right)}{(2\pi)^{l/2}|\mathbf{W}|^{1/2}}\,,$$

which can be summarised as

$$\mathbf{u}|\mathbf{y} \sim \mathcal{N}\left(\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}),\mathbf{W}^{-1}\right)\,.$$

As such, the best predictor $\tilde{\mathbf{u}}$ of $\mathbf{u}$ is:

$$\tilde{\mathbf{u}} = \mathrm{E}(\mathbf{u}|\mathbf{y}) = \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})\,.$$

If we want this predictor to be linear in $\mathbf{y}$ and unbiased, a linear and unbiased

estimator of $\mathbf{X}\boldsymbol{\beta}$ is required. Such an estimator is already known as the GLSE of $\mathbf{X}\boldsymbol{\beta}$. The best linear unbiased estimators/predictors of respectively $\boldsymbol{\beta}$ and $\mathbf{u}$ can thus be summarised as:

$$\text{BLUE}(\boldsymbol{\beta}) = \hat{\boldsymbol{\beta}} = \left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y} \tag{3.8}$$

$$\text{BLUP}(\mathbf{u}) = \hat{\mathbf{u}} = \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}})\,. \tag{3.9}$$

Although these estimators are in theory unbiased, a bad estimation of the covariance structure $\mathbf{V}$, might result in an estimator/predictor that is no longer unbiased. Therefore, it is essential to estimate the variance components as correct as possible based on the entire data set.

3.3. Mixed Model Equations (MME)

Before it was shown that Eq. (3.8) and (3.9) yield the estimators/predictors that minimise the mean squared error and are unbiased, Henderson already found these estimators/predictors by searching for the maximum likelihood estimates for the fixed effects of a linear mixed model [48]. The system of equations leading to these estimators/predictors was derived by maximising the joint density function of $\mathbf{y}$ and $\mathbf{u}$, but the form of this joint density function was slightly different than the one we proposed in Eq. (3.5). An equivalent formula for the joint density function can be found by using the Bayes theorem of Eq. (3.4) in a different form:

$$f(\mathbf{y}, \mathbf{u}) = f(\mathbf{y}|\mathbf{u})f(\mathbf{u})\,,$$

with $f(\mathbf{u})$ the marginal density function of $\mathbf{u}$,

$$f(\mathbf{u}) = \frac{\exp\left(-\frac{1}{2}\mathbf{u}'\mathbf{G}^{-1}\mathbf{u}\right)}{(2\pi)^{l/2}|\mathbf{G}|^{1/2}}\,,$$

and $f(\mathbf{y}|\mathbf{u})$ is the conditional density function of $\mathbf{y}$ given $\mathbf{u}$, recalling that $\text{var}(\mathbf{e}) = \mathbf{R}$:

$$f(\mathbf{y}|\mathbf{u}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u})'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta} - \mathbf{Z}\mathbf{u})\right)}{(2\pi)^{n/2}|\mathbf{R}|^{1/2}}\,.$$

The joint density function can then be written as:

$$f(\mathbf{y},\mathbf{u}) = \frac{\exp\left(-\frac{1}{2}\left((\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})'\mathbf{R}^{-1}(\mathbf{y}-\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}\mathbf{u})+\mathbf{u}'\mathbf{G}^{-1}\mathbf{u}\right)\right)}{(2\pi)^{(n+l)/2}|\mathbf{R}|^{1/2}|\mathbf{G}|^{1/2}}\,.$$

The attentive reader will notify the equality between this expression and Eq. (3.5), which can be derived using some algebraic tricks. Maximisation of this joint density function with respect to $\boldsymbol{\beta}$ and $\mathbf{u}$ is done by taking the partial derivatives with respect to these variables of $f(\mathbf{y},\mathbf{u})$,

$$\begin{aligned}\frac{\partial f(\mathbf{y},\mathbf{u})}{\partial \boldsymbol{\beta}} &= \left(\mathbf{X}'\mathbf{R}^{-1}\mathbf{y}-\mathbf{X}'\mathbf{R}^{-1}\mathbf{X}\boldsymbol{\beta}-\mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\mathbf{u}\right) f(\mathbf{y},\mathbf{u})\,,\\ \frac{\partial f(\mathbf{y},\mathbf{u})}{\partial \boldsymbol{\beta}} &= \left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{y}-\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}\boldsymbol{\beta}-\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}\mathbf{u}+\mathbf{G}^{-1}\mathbf{u}\right) f(\mathbf{y},\mathbf{u})\,,\end{aligned}$$

and equating these partial derivatives to zero. Denoting the solutions for $\boldsymbol{\beta}$ and $\mathbf{u}$ by $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{u}}$, yields the following equations:

$$\mathbf{X}'\mathbf{R}^{-1}\mathbf{X}\hat{\boldsymbol{\beta}}+\mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\hat{\mathbf{u}} = \mathbf{X}'\mathbf{R}^{-1}\mathbf{y}\,, \tag{3.10}$$

$$\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}\hat{\boldsymbol{\beta}}+(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1})\hat{\mathbf{u}} = \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y}\,. \tag{3.11}$$

These are known as the mixed model equations (MME) and can be written more compactly as

$$\begin{bmatrix}\mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\beta}}\\ \hat{\mathbf{u}}\end{bmatrix} = \begin{bmatrix}\mathbf{X}'\mathbf{R}^{-1}\mathbf{y}\\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y}\end{bmatrix} \tag{3.12}$$

It will now be shown that the solutions of these mixed model equations are exactly the same as the BLUE for $\boldsymbol{\beta}$ and the BLUP for $\mathbf{u}$. Therefore, we will first rewrite Eq. (3.11) as

$$\hat{\mathbf{u}} = (\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1})^{-1}\mathbf{Z}'\mathbf{R}^{-1}(\mathbf{y}-\mathbf{X}\hat{\boldsymbol{\beta}}) \tag{3.13}$$

and substitute this in Eq. (3.10)

$$\mathbf{X}'\left(\mathbf{R}^{-1}-\mathbf{R}^{-1}\mathbf{Z}(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1})^{-1}\mathbf{Z}'\mathbf{R}^{-1}\right)(\mathbf{X}\hat{\boldsymbol{\beta}}-\mathbf{y}) = \mathbf{0}\,. \tag{3.14}$$

From the blockwise inversion of a matrix where both diagonal blocks are

invertible, we can deduce that

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}$$

and substituting $\mathbf{D}$ by $-\mathbf{D}$ gives us

$$(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}\,.$$

Using this result in Eq. (3.14) transforms this equation into

$$\mathbf{X}'(\mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}')^{-1}(\mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{y}) = \mathbf{0}\,.$$

With $\mathbf{V} = \mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}'$ as in Eq. (3.2), the solution of this equation is:

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}\,,$$

which is exactly the BLUE for $\boldsymbol{\beta}$ as in Eq. (3.8).

To find the solution for $\hat{\mathbf{u}}$, we will again use a result from the blockwise inversion of a matrix where both diagonal blocks are invertible:

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} = \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}$$

and again when substituting $\mathbf{D}$ by $-\mathbf{D}$:

$$(\mathbf{A} + \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}\mathbf{B}\mathbf{D}^{-1} = \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} + \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\,.$$

This result can be applied on Eq. (3.11) as formatted in Eq. (3.13):

$$\begin{aligned}\hat{\mathbf{u}} =& (\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1})^{-1}\mathbf{Z}'\mathbf{R}^{-1}(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}) \\ =& \mathbf{G}\mathbf{Z}'(\mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}')^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \\ =& \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\,,\end{aligned}$$

leading to the BLUP equation as in Eq. (3.9). The mixed model equations are thus a practical tool for finding BLUP and BLUE for the random and fixed effects in a linear mixed model when the covariance matrices are known. However, the latter is commonly not true and as already mentioned, the covariance matrices mostly depend on some variance components that should be estimated based on the data. This is the subject of the next chapter, where especially Restricted Maximum Likelihood will be discussed for estimating variance components.

3.4. Computational aspects of breeding applications

The greatest accomplishment of the mixed model equations is the fact that the covariance matrix $\mathbf{V}$ of $\mathbf{y}$ is no longer needed to be calculated and inverted to find BLUE and BLUP of $\boldsymbol{\beta}$ and $\mathbf{u}$. Only the inversion of $\mathbf{G}$ and $\mathbf{R}$ is required, which were commonly assumed to be diagonal and thus easily invertible. However, problems arose when breeders wanted to incorporate pedigree information by inserting a relationship matrix as covariance matrix for the random genetic effects. Such a relationship matrix was usually not diagonal and at a time when no personal computers were yet available, the calculation of an inverse of a non-diagonal matrix was a tedious and time-consuming job. As the mixed model equations needed the inverse of the covariance matrix of the genetic effects, Henderson in 1976 came up with an inventive method to directly construct the inverse of the relationship matrix, without needing to explicitly compute the relationship matrix [45]. This method was extremely useful when the base population was unrelated, non-inbred and not selected. If these assumptions were not fulfilled, it could lead to a singular relationship matrix, making it impossible to calculate the inverse of the relationship matrix. A solution for such a problem was proposed by Harville, leading to a new symmetric system, without the need for inverting $\mathbf{G}$ [49]:

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\mathbf{G} \\ \mathbf{G}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{G}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}\mathbf{G}+\mathbf{G} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{d}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{G}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}, \tag{3.15}$$

where the BLUP for the original random effects $\mathbf{u}$ are found as $\hat{\mathbf{u}} = \mathbf{G}\hat{\mathbf{d}}$. An important remark here is that due to singularity of $\mathbf{G}$, the coefficient matrix of these transformed mixed model equations is also singular and thus the solution for $\mathbf{d}$ is not unique, but $\hat{\mathbf{u}} = \mathbf{G}\hat{\mathbf{d}}$ is nonetheless a unique solution [49].

The use of SNP marker effects in breeding applications dramatically increased the number of random effects included in the model and thus also the dimensionality of the mixed model equations. Also, when using marker effects as random effects, the covariance matrix for the random effects was mainly assumed to be a constant diagonal matrix of the form $\mathbf{G} = \sigma_u^2\mathbf{I}$ and thus initially, the problems with inverting $\mathbf{G}$ were no longer applicable.

However, larger problems that did not make use of marker effects could benefit from the fact that $\mathbf{Z}$ was a very sparse matrix, because the random effects were genetic effects specific for a certain animal and there were commonly only a few records of the same animal in a data set. Furthermore, the relationship matrix, which was used as the covariance matrix $\mathbf{G}$ in these models was mostly also sparse in large data sets, due to the fact that in large populations many animals are unrelated. This led to a sparse coefficient matrix that can be stored very efficiently and for which iterative techniques can be optimised to solve the mixed model equations very rapidly.

Although in genomic prediction with random SNP marker effects, the covariance matrix $\mathbf{G}$ is also sparse, the incidence matrix $\mathbf{Z}$ is not. Even if the allele coding was chosen such that homozygosity in the most frequent allele was coded as a zero, the multplication $\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}$ always results in an almost completely dense coefficient matrix. In fact, the common assumption is $\mathbf{R} = \sigma_e^2\mathbf{I}$ and thus the only zero elements in $\mathbf{Z}'\mathbf{Z}$ will be due to alleles that are in complete linkage disequilibrium. The introduction of SNP marker effects as random effects thus leads to higher memory requirements if the coefficient matrix should be stored explicitly. Legarra compared several strategies to deal with such mixed model equations and came to the conclusion that iterative techniques that do not require the complete storage of the coefficient matrix, such as preconditioned conjugate gradient (PCG) [50] and Gauss-Seidel iterations [51], are preferred for solving the mixed model equations [35]. However, these iterative techniques, also known as iteration on data methods, may have problems with numerical stability and convergence when the coefficient matrix is ill-conditioned. An important side note is the fact that the time to solve for these iterative techniques is proportional to the number of records, just as the memory requirements. On the other hand, for a direct solution of such a matrix equation, for example using Cholesky decomposition, the time to solve and the memory requirements are constant for a given number of effects. The iteration on data methods also do not set up the entire mixed model equations, but process the incidence matrices line per line from file each iteration [52], which might result in an important increase in computing time for large-scale data sets.

To reduce numerical problems with PCG or GS when using marker data explicitly as random effects, VanRaden used the SNP marker data to construct a so-called genomic relationship matrix which could be used instead of the relationship matrix based on the pedigree as the covariance matrix $\mathbf{G}$ [16].

However, this genomic relationship matrix could be singular, again leading to the use of the transformed Eq. (3.15), or a weighted mean of the pedigree and genomic relationship matrix could be chosen to avoid singularity. But the creation of these genomic relationship matrices is quite costly and quadratically proportional to the number of genotyped individuals included. Even dedicated implementations needed about three hours to construct a genomic relationship matrix for 30,000 individuals, genotyped with 40,000 markers [53].

It was already mentioned earlier on that some widely used assumptions are:

$$\text{var}(\mathbf{e}) = \mathbf{R} = \sigma_e^2 \mathbf{I}\,, \qquad \text{var}(\mathbf{u}) = \mathbf{G} = \sigma_u^2 \mathbf{I}\,,$$

leading to the simplified mixed model equations

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} + \frac{\sigma_e^2}{\sigma_u^2}\mathbf{I} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{y}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{Z}'\mathbf{y} \end{bmatrix}\,,$$

which can also be written as

$$(\mathbf{W}'\mathbf{W} + \lambda^2 \mathbf{D})\boldsymbol{\alpha} = \mathbf{W}'\mathbf{y}\,,$$

with

$$\mathbf{W} = \begin{bmatrix} \mathbf{X} & \mathbf{Z} \end{bmatrix}\,, \qquad \mathbf{D} = \begin{bmatrix} \mathbf{0}_{k\times k} & \mathbf{0}_{k\times l} \\ \mathbf{0}_{l\times k} & \mathbf{I}_l \end{bmatrix}\,, \qquad \lambda^2 = \frac{\sigma_e^2}{\sigma_u^2}\,,$$

where $\mathbf{I}_l$ is a square identity matrix with as dimension the number of random effects, $\mathbf{0}_{i\times j}$ is a zero matrix with as dimension $i \times j$, k and l are resp. the numbers of fixed and random effects. This is very alike to a ridge regression formulation and therefore, this specific form of the mixed model equations is mainly referred to as Ridge Regression BLUP (RR-BLUP) [54]. Piepho reviewed 6 methods to fit this RR-BLUP model in a computationally efficient way [55]. As a conclusion, the most efficient method was the one using a spectral decomposition as proposed by Kang et al. [56]. However, it is almost impossible to extend this model to incorporate other effects and variances, which makes it impractical for use in realistic scenarios. Furthermore, this method and most other methods described by Piepho were tested on data sets where the number of effects is a lot larger than the number of genotyped individuals, benefiting methods that use a genomic relationship matrix instead of explicitly modelling the marker

effects. Nonetheless, due to the ever decreasing cost of genotyping, the number of genotyped individuals will most probably surpass the number of SNP markers, making it computationally more efficient to model these SNP markers explicitly.

4 Variance component estimation with Restricted Maximum Likelihood

4.1. Introduction

In the previous chapter we have assumed that the covariance matrices of the random effects and the residuals were known. However, in reality only the covariance structure is assumed known, but it still depends on some variance components that need to be estimated based on the data. Again, this chapter will focus on the specific methods and algorithms that have been chosen as the most appropriate for this dissertation. Readers with a broader interest in variance component estimation are recommended to consult the book "Variance Components" by Searle, which gives a general overview of variance component estimation procedures from different perspectives [43].

Variance component estimation has always played a major role in quantitative genetics and some major contributions in this field actually came from solving problems that had emerged in an animal breeding context. Actually, in 1919 Fisher initiated one of the first methods for variance component estimation called the analysis of variance method (ANOVA) in his landmark paper about the analysis of human variability through Mendelian inheritance [6]. The ANOVA method has been widely used from the 1940s and still is commonly used today, although it is very simple in nature and initially had some flaws, which needed to be overcome to make it more useful in practice. The simplicity of the ANOVA method originates from the fact that it equates the expected values of the variance components to their so-called sums of squares. If we recall the model used in the previous chapter for analysing clinical trials, the effect of a single treatment can be inferred from a simple random effects model

$$y_{ij} = \mu + u_i + e_{ij} \,,$$

with y_{ij} the effect of the treatment on each patient j, μ the mean effect of the treatment, u_i the random effect of clinic i and e_{ij} the residual effect

of each patient. When the treatment is applied in a clinics on n patients per clinic, it is called a balanced model with a classes of the random effects. Assuming $\text{var}(u_i) = \sigma_u^2$, $\text{var}(e_{ij}) = \sigma_e^2$ and all covariances equal to zero, the ANOVA estimates for the variance components in such a balanced random effects model can be summarised as:

$$\begin{aligned}\hat{\sigma}_e^2 =& \frac{1}{a(n-1)} \sum_{i=1}^{a} \sum_{j=1}^{n} (y_{ij} - \bar{y}_{i\cdot})^2 \,, \\ \hat{\sigma}_u^2 =& \frac{1}{a-1} \sum_{i=1}^{a} (\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2 - \frac{\hat{\sigma}_e^2}{n} \,, \end{aligned} \tag{4.1}$$

with

$$\bar{y}_{i\cdot} = \frac{1}{n} \sum_{j=1}^{n} y_{ij} \qquad \text{and} \qquad \bar{y}_{\cdot\cdot} = \frac{1}{a} \sum_{i=1}^{a} \bar{y}_{i\cdot} \,.$$

Of course, many adaptations and extensions of this method have been studied, but a particularly interesting adaptation was motivated by the problem in dairy cattle breeding of how to use unbalanced data for variance component estimation. As can be seen in Eq. (4.1) the original ANOVA equations make explicit use of the balancedness of the data by the presence of n in the denominators. In 1953 Henderson developed 3 methods that were adaptations of the ANOVA method for using unbalanced data from random or mixed models for the estimation of the variance components [7]. We will not go into the details of these methods, but method I is the computationally most practical method by just using analogous formulas as the ANOVA estimates, but with denominators that depend on the number of records in each class of the random effects. Method II is more specifically aimed at mixed models, because fixed effects are first estimated by their least squares estimates and the observations are then adjusted for these fixed effects. Subsequently, Method I is applied on these adjusted observations for the estimation of the variance components of the random effects. Method III is based on using sums of squares where the average values in Eq. (4.1) are replaced by estimated values for y after fitting a linear model, yielding unbiased estimates and being the most satisfactory method of all 3. However, Henderson himself already noted that this method quickly became computationally unfeasible when the number of different classes for the random effects became high.

Nevertheless, some other weaknesses that are inherent to the ANOVA method remained unsolved by Henderson. In particular, estimates for the variance

components could still be negative as can be seen in Eq. (4.1) where $\hat{\sigma}_u^2$ can become negative whenever $\hat{\sigma}_e^2$ exceeds $\frac{n}{a-1}\sum_{i=1}^{a}(\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2$. Another negative aspect of the ANOVA method is the fact that no prior information about the distributional properties can be included in the estimation process. These problems were mainly solved by the introduction of Maximum Likelihood (ML) estimators for the variance components. It were Hartley and Rao who developed the methodology for a very wide class of models in 1967 [57]. Although Fisher already derived the method of maximum likelihood in 1922, it was commonly rejected as an estimation method due to the computational effort connected to the maximum likelihood estimation [58]. Hartley and Rao circumvented this problem by applying matrix algebra and advanced numerical methods to alleviate the computational burden.

The solutions of the maximum likelihood estimation are mostly not available in a closed analytical form, which means that iterative techniques should be used to obtain estimates of the variance components. This was one of the reasons why, initially, ML estimation was not used as widespread as ANOVA methods, but with the advent of new computing methods and more powerful computers, the main problems inherent to maximum likelihood estimation were circumvented and ML became an attractive variance component estimation procedure. Nonetheless, for balanced data, maximum likelihood estimators are not equal to ANOVA estimators, which have the attractive property of being unbiased. Therefore, Patterson and Thompson considered the maximum likelihood estimation of that part of the likelihood that was invariant to the fixed effects [59]. This has become known as restricted or residual maximum likelihood (REML). The computational burden of REML is identical to that of ML estimation, but REML has the beneficial property that it takes into account the number of degrees of freedom used for estimating fixed effects when estimating variance components. Another advantage is that for balanced data, the REML estimates are equal to the ANOVA estimates and thus unbiased. Therefore, REML estimation will be discussed more thoroughly in the following section as it is the preferred method of estimation, especially for unbalanced data.

In the early days of variance component estimation, the covariance structures were kept as simple as possible and were mainly considered to be constant diagonal matrices. However, during the last decades, more complex covariance structures have been used with an ever increasing number of variance

components to be estimated. The advantage of REML estimation is the fact that it can be used for any covariance structure and any number of variance components, but of course the complexity of solving the REML estimation equations will increase with increasing complexity of the covariance structure and with a higher number of variance components. Although in the previous chapter we argued that a simple covariance structure could suffice for genomic selection, a more general derivation of the REML estimation procedure will be given in the next section to show its usefulness for more complex models.

The remainder of this chapter is organised as follows:

- Section 4.2 will derive the REML estimation equations for variance component estimation in a very general case.
- Section 4.3 will present an iterative algorithm called the Average Information algorithm (AI-REML), which has a convenient computational way of maximizing the REML likelihood function.
- Section 4.4 will review the computational aspects of applying the AI-REML algorithm and will review other commonly used iterative techniques for maximising the REML likelihood function in a breeding perspective, while comparing them with the AI-REML algorithm.

In this chapter, many results from linear algebra are used, and some of the derivations can be found in the Appendix. However, readers who are interested in the details of these algebraic results are referred to the book "Matrix algebra from a statistician's perspective" by Harville [60].

4.2. Restricted Maximum Likelihood (REML)

The starting point of the derivation of the Restricted Maximum Likelihood equations is the linear mixed model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}\,,$$

where $\mathbf{y}$ is a vector of n observations, $\boldsymbol{\beta}$ is the vector of k fixed effects with incidence matrix $\mathbf{X}$, $\mathbf{u}$ is the vector of l random effects with incidence matrix $\mathbf{Z}$ and $\mathbf{e}$ is the vector of n residuals. In the following we will always assume that $\mathbf{X}$ and $\mathbf{Z}$ are full of rank, even though this is not a necessary

condition. The assumptions for the distributions of the random variables and the residuals can be summarised as follows:

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \begin{bmatrix} \mathbf{G}(\boldsymbol{\gamma}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}(\boldsymbol{\phi}) \end{bmatrix} \right),$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$ are vectors of variance components on which the covariance matrices rely. For the moment, we will make no further assumptions on the structure of the covariance matrices to keep this derivation as general as possible. The probability density function of

$$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{V}),$$

with

$$\mathbf{V} = \mathbf{Z}\mathbf{G}(\boldsymbol{\gamma})\mathbf{Z}' + \mathbf{R}(\boldsymbol{\phi}),$$

can then be written as

$$f(\mathbf{y}) = \frac{\exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right)}{(2\pi\sigma^2)^{n/2}|\mathbf{V}|^{1/2}}.$$

The likelihood function for the parameters $\boldsymbol{\beta}$, σ^2, $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$ is exactly this probability density function:

$$L = L(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{y}) = \frac{\exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})\right)}{(2\pi\sigma^2)^{n/2}|\mathbf{V}|^{1/2}}.$$

Maximisation of this likelihood is usually performed by maximising the logarithm of the likelihood function, the so-called log-likelihood function

$$\begin{aligned} l(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{y}) &= \log(L) \\ &= -\frac{1}{2}\Big(n\log(2\pi\sigma^2) + \log|\mathbf{V}| \\ &\quad + \frac{1}{\sigma^2}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})'\mathbf{V}^{-1}(\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) \Big). \end{aligned} \tag{4.2}$$

Maximising in function of $\boldsymbol{\beta}$ will result in a maximum likelihood estimator for $\boldsymbol{\beta}$, which is obtained by taking the derivative of $l(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{y})$ with respect to $\boldsymbol{\beta}$

$$\frac{\partial l(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{y})}{\partial \boldsymbol{\beta}} = \frac{1}{\sigma^2}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{y} - \mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\boldsymbol{\beta}\right)$$

and equating it to zero, leading to the maximum likelihood estimator for $\boldsymbol{\beta}$, denoted as $\tilde{\boldsymbol{\beta}}$,

$$\tilde{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}\,. \tag{4.3}$$

This estimator is exactly the same as the BLUE from the previous chapter.

Maximising the log-likelihood in Eq. (4.2) with respect to the variance components would lead to ML estimators for these variance components. However, the restricted maximum likelihood (REML) estimators maximise that part of the likelihood function that is invariant to the fixed effects. Therefore, linear combinations of elements of $\mathbf{y}$ are chosen such that they are no longer dependent on $\boldsymbol{\beta}$. This was referred to by Harville as an error-contrast , with zero as its expected value [61]. A set of these error-contrasts $\mathbf{k}'\mathbf{y}$ can be found by solving the following equation for $\mathbf{k}'$

$$\mathrm{E}(\mathbf{k}'\mathbf{y}) = \mathbf{k}'\mathbf{X}\boldsymbol{\beta} = \mathbf{0}\,.$$

As this equation should hold for any $\boldsymbol{\beta}$, a general form for $\mathbf{k}'$ can be found as the solution of

$$\mathbf{k}'\mathbf{X} = \mathbf{0}\,,$$

which can easily be verified to be

$$\mathbf{k}' = \mathbf{c}'\left(\mathbf{I}_n - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\right)\,,$$

for any $\mathbf{c}'$ and with $\mathbf{I}_n$ the identity matrix of dimension n. We can thus find an infinite number of these error-contrasts, but as we are only interested in a set of linearly independent error-contrasts, it is first checked how many of these linearly independent error-contrasts exist.

The matrix $\mathbf{T} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ is an idempotent matrix, which means that $\mathbf{TT} = \mathbf{T}$:

$$\mathbf{TT} = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' = \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' = \mathbf{T}\,.$$

Two important properties of idempotent matrices are firstly that if $\mathbf{T}$ is an idempotent matrix, then $\mathbf{I} - \mathbf{T}$ is also an idempotent matrix; and secondly that the trace of an idempotent matrix is equal to its rank. Therefore, the

rank of $\mathbf{I}_n - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$ can be calculated as

$$\begin{aligned}\operatorname{rank}(\mathbf{I}_n - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') &= \operatorname{tr}(\mathbf{I}_n - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') \\ &= \operatorname{tr}(\mathbf{I}_n) - \operatorname{tr}(\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') \\ &= n - \operatorname{tr}(\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}) \\ &= n - \operatorname{tr}(\mathbf{I}_k) \\ &= n - k\,.\end{aligned}$$

For the last step we have used the property that $\operatorname{tr}(\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') = \operatorname{tr}(\mathbf{X}'\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}) = \operatorname{tr}(\mathbf{I}_k)$. From this result we can conclude that there are no more than $n-k$ linearly independent vectors $\mathbf{k}'$. If we use such a set of linearly independent vectors as the rows of a matrix $\mathbf{K}'$, then the REML equations are based on maximising the likelihood of

$$\mathbf{K}'\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{K}'\mathbf{V}\mathbf{K})\,,$$

where $\mathbf{K}'$ has full row rank $n-k$.

One may wonder why we only need $n-k$ linear combinations of the total of n observations to estimate the variance components and if we don't loose information by not using all observations. An intuitive explanation is the fact that k linear combinations of observations have been used for an ML estimation of the fixed effects, while $n-k$ linearly independent combinations of observations will be used for a REML estimation of the variance components. This has also been proven by many others by showing that inferences based on $\mathbf{y}$ can also be obtained by maximising the likelihood of a transposed data vector $\mathbf{A}\mathbf{y}$, which can be split into maximising a likelihood that depends on $\boldsymbol{\beta}$ and one that does not depend on $\boldsymbol{\beta}$ [59, 61, 62]. Maximising the likelihood that depends on $\boldsymbol{\beta}$ with respect to $\boldsymbol{\beta}$ results in the ML estimation for $\boldsymbol{\beta}$ as shown in Eq. (4.3), while maximising the likelihood independent of $\boldsymbol{\beta}$ comes down to maximising the likelihood of $\mathbf{K}'\mathbf{y}$.

The probability density function of $\mathbf{K}'\mathbf{y}$ is

$$f(\mathbf{K}'\mathbf{y}) = \frac{\exp\left(-\frac{1}{\sigma^2}\mathbf{y}'\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{y}\right)}{(2\pi\sigma^2)^{(n-k)/2}|\mathbf{K}'\mathbf{V}\mathbf{K}|^{1/2}}$$

and thus we can write the REML log-likelihood function as

$$l_{\text{REML}}(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{K}'\mathbf{y}) = -\frac{1}{2}\left((n-k)\log(2\pi\sigma^2) + \log|\mathbf{K}'\mathbf{V}\mathbf{K}| + \frac{\mathbf{y}'\mathbf{K}\left(\mathbf{K}'\mathbf{V}\mathbf{K}\right)^{-1}\mathbf{K}'\mathbf{y}}{\sigma^2}\right). \tag{4.4}$$

This log-likelihood can be made independent of $\mathbf{K}$ by applying some algebraic tricks. First we will focus on the last term of Eq. (4.4). To that end, we will construct the matrix $\mathbf{T} = \mathbf{I}_n - \mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2} - \mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2}$ and show that this matrix is a null matrix. This can be done by first showing that $\mathbf{T}$ is idempotent:

$$\begin{aligned}
\mathbf{T}\mathbf{T} =& \left(\mathbf{I}_n - \mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2} - \mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2}\right)^2 \\
=& \mathbf{I}_n - 2\mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2} - 2\mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2} \\
&+ \left(\mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2}\right)^2 + \left(\mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2}\right)^2 \\
&+ \mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2}\mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2} \\
&+ \mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2}\mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2} \\
=& \mathbf{I}_n - \mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2} - \mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2} \\
=& \mathbf{T}\,,
\end{aligned} \tag{4.5}$$

where we have made use of the fact that $\mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2}$ and $\mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2}$ are idempotent matrices and $\mathbf{K}'\mathbf{X} = \mathbf{0} = \mathbf{X}'\mathbf{K}$. We know that for an idempotent matrix its rank is equal to its trace and thus

$$\begin{aligned}
\text{rank}(\mathbf{T}) =& \text{tr}(\mathbf{T}) \\
=& \text{tr}(\mathbf{I}_n) - \text{tr}\left(\mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2}\right) \\
&- \text{tr}\left(\mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2}\right) \\
=& n - \text{tr}(\mathbf{I}_k) - \text{tr}(\mathbf{I}_{n-k}) \\
=& n - k - (n-k) = 0\,,
\end{aligned} \tag{4.6}$$

where we have made use of the property of traces that $\text{tr}(\mathbf{ABC}) = \text{tr}(\mathbf{CAB})$. The only matrix that has rank zero is the zero matrix and thus we have

proven that $\mathbf{T} = \mathbf{0}$. As such,

$$\mathbf{V}^{1/2}\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{1/2} = \mathbf{I}_n - \mathbf{V}^{-1/2}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1/2}$$

and multiplication to the left and to the right by $\mathbf{V}^{-1/2}$ results in:

$$\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}' = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1} = \mathbf{P}\,,$$

where we have introduced the matrix $\mathbf{P}$ because it will be used frequently later on and it has some special properties that will also be described when necessary. The REML log-likelihood is thus reduced to

$$l_{\mathrm{REML}}(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{K}'\mathbf{y}) = -\frac{1}{2}\left((n-k)\log(2\pi\sigma^2) + \log|\mathbf{K}'\mathbf{V}\mathbf{K}| + \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^2}\right).$$

Let us now focus on the determinant of $\mathbf{K}'\mathbf{V}\mathbf{K}$ and construct first the square $n \times n$ matrix

$$\mathbf{L} = \begin{bmatrix}\mathbf{K} & \mathbf{V}^{-1}\mathbf{X}\end{bmatrix}.$$

It can be shown that this matrix is non-singular by proving that the determinant of $\mathbf{L}'\mathbf{L}$ is not zero:

$$\begin{aligned}\left|\mathbf{L}'\mathbf{L}\right| &= \det\left(\begin{bmatrix}\mathbf{K}'\mathbf{K} & \mathbf{K}'\mathbf{V}^{-1}\mathbf{X} \\ \mathbf{X}'\mathbf{V}^{-1}\mathbf{K} & \mathbf{X}'\mathbf{V}^{-2}\mathbf{X}\end{bmatrix}\right) \\ &= \left|\mathbf{K}'\mathbf{K}\right|\left|\mathbf{X}'\mathbf{V}^{-2}\mathbf{X} - \mathbf{X}'\mathbf{V}^{-1}\mathbf{K}(\mathbf{K}'\mathbf{K})^{-1}\mathbf{K}'\mathbf{V}^{-1}\mathbf{X}\right| \\ &= \left|\mathbf{K}'\mathbf{K}\right|\left|\mathbf{X}'\mathbf{V}^{-1}\left(\mathbf{I}_n - \mathbf{K}(\mathbf{K}'\mathbf{K})^{-1}\mathbf{K}'\right)\mathbf{V}^{-1}\mathbf{X}\right|.\end{aligned}$$

Analogously to Eq. (4.5) and (4.6), it can be shown that $\mathbf{I_n} - \mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}' - \mathbf{K}(\mathbf{K}'\mathbf{K})^{-1}\mathbf{K}'$ is a null matrix and so

$$\begin{aligned}\left|\mathbf{L}'\mathbf{L}\right| &= \left|\mathbf{K}'\mathbf{K}\right|\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| \\ &= \left|\mathbf{K}'\mathbf{K}\right|\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|\left|\mathbf{X}'\mathbf{X}\right|^{-1}\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|.\end{aligned}$$

None of the matrices on the right-hand side are singular leading to the conclusion that $\mathbf{L}'\mathbf{L}$ and thus also $\mathbf{L}$ are not singular. If we now look at the

determinant of $\mathbf{L}'\mathbf{V}\mathbf{L}$, then

$$\begin{aligned}\left|\mathbf{L}'\mathbf{V}\mathbf{L}\right| &= \left|\mathbf{K}'\mathbf{V}\mathbf{K}\right| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X} - \mathbf{X}'\mathbf{K}(\mathbf{K}'\mathbf{V}\mathbf{K})^{-1}\mathbf{K}'\mathbf{X}\right| \\ &= \left|\mathbf{K}'\mathbf{V}\mathbf{K}\right| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X} - \mathbf{X}'\mathbf{P}\mathbf{X}\right| \\ &= \left|\mathbf{K}'\mathbf{V}\mathbf{K}\right| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| ,\end{aligned}$$

where we have used a first property of $\mathbf{P}$, namely that

$$\begin{aligned}\mathbf{P}\mathbf{X} &= \mathbf{V}^{-1}\mathbf{X} - \mathbf{V}^{-1}\mathbf{X}(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X})^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{X} \\ &= \mathbf{V}^{-1}\mathbf{X} - \mathbf{V}^{-1}\mathbf{X} = \mathbf{0} .\end{aligned}$$

Making use of the property that the determinant of a product of square matrices of equal dimensions is equal to the product of their determinants, the determinant of $\mathbf{K}'\mathbf{V}\mathbf{K}$ can be replaced by:

$$\begin{aligned}\left|\mathbf{K}'\mathbf{V}\mathbf{K}\right| &= \left|\mathbf{L}'\mathbf{V}\mathbf{L}\right| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|^{-1} \\ &= \left|\mathbf{L}'\mathbf{L}\right| |\mathbf{V}| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|^{-1} \\ &= \left|\mathbf{K}'\mathbf{K}\right| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|^{2} \left|\mathbf{X}'\mathbf{X}\right| |\mathbf{V}| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|^{-1} \\ &= \left|\mathbf{K}'\mathbf{K}\right| \left|\mathbf{X}'\mathbf{X}\right| |\mathbf{V}| \left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| ,\end{aligned}$$

where the two first determinants are independent of the variance components. The REML log-likelihood can then be rewritten by omitting constant factors as:

$$\begin{aligned}l_{\mathrm{REML}}(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}|\mathbf{K}'\mathbf{y}) = -\Bigg(&(n-k)\log(\sigma^2) + \log\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| \\ &+ \log|\mathbf{V}| + \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^2}\Bigg) . \end{aligned} \tag{4.7}$$

This equation can be transformed so that the dependence on the different variance components becomes more clear by looking at the mixed model equations as derived in the previous chapter

$$\begin{bmatrix}\mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1}\end{bmatrix}\begin{bmatrix}\hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}}\end{bmatrix} = \begin{bmatrix}\mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y}\end{bmatrix} .$$

Computing the determinant of the coefficient matrix $\mathbf{C}$ of this system of

equations leads to

$$\begin{aligned}|\mathbf{C}| &= \left|\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right|\left|\mathbf{X}'\mathbf{R}^{-1}\mathbf{X}-\mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1})^{-1}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}\right| \\ &= \left|\mathbf{G}^{-1}(\mathbf{I}_l+\mathbf{G}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z})\right|\left|\mathbf{X}'\left(\mathbf{R}^{-1}-\mathbf{R}^{-1}\mathbf{Z}(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1})^{-1}\mathbf{Z}'\mathbf{R}^{-1}\right)\mathbf{X}\right| .\end{aligned}$$

We can again exploit the property of blockwise inversion of a matrix with 2 invertible diagonal blocks $\mathbf{A}$ and $\mathbf{D}$

$$(\mathbf{A}+\mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}=\mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{B}(\mathbf{D}+\mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}$$

and Sylvester's determinant theorem for matrix $\mathbf{A}$ of dimension $m \times n$ and $\mathbf{B}$ of dimension $n \times m$ [63]:

$$|\mathbf{I}_m+\mathbf{A}\mathbf{B}|=|\mathbf{I}_n+\mathbf{B}\mathbf{A}| .$$

This leads to

$$\begin{aligned}|\mathbf{C}| &= |\mathbf{G}|^{-1}\left|\mathbf{I}_n+\mathbf{R}^{-1}\mathbf{Z}\mathbf{G}\mathbf{Z}'\right|\left|\mathbf{X}'(\mathbf{R}+\mathbf{Z}\mathbf{G}\mathbf{Z}')^{-1}\mathbf{X}\right| \\ &= |\mathbf{G}|^{-1}\left|\mathbf{R}^{-1}(\mathbf{R}+\mathbf{Z}\mathbf{G}\mathbf{Z}')\right|\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| \\ &= |\mathbf{G}|^{-1}|\mathbf{R}|^{-1}|\mathbf{V}|\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right| .\end{aligned}$$

Taking the logarithm of this expression and moving some terms to the other side leads to

$$\log|\mathbf{V}|+\log\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|=\log|\mathbf{C}|+\log|\mathbf{G}|+\log|\mathbf{R}|$$

and hence the REML log-likelihood can be rewritten as

$$\begin{aligned}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi}|\mathbf{K}'\mathbf{y}) = -\Bigg((n-k)\log(\sigma^2)+\log|\mathbf{C}|+\log|\mathbf{G}| \\ +\log|\mathbf{R}|+\frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^2}\Bigg) .\end{aligned} \tag{4.8}$$

The maximisation of this REML log-likelihood function is performed by derivation of this function with respect to the different variance components and equating these derivatives to zero. The REML estimate for σ^2 can thus be found by equating to zero

$$\frac{\partial l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\sigma^2}=-\left(\frac{n-k}{\sigma^2}-\frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^4}\right) . \tag{4.9}$$

An analytical solution can thus be found for the REML estimate of σ^2 in the form of

$$\hat{\sigma}^2 = \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{n-k}\,.$$

Most often, this is the only variance component for which an analytical expression of its REML estimate can be found.

For the variance components γ_i included in the vector $\boldsymbol{\gamma}$, the derivation of the REML estimates is a bit more tedious:

$$\frac{\partial l_{\mathrm{REML}}(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi})}{\partial \gamma_i} = -\left(\frac{\partial \log|\mathbf{C}|}{\partial \gamma_i} + \frac{\partial \log|\mathbf{G}|}{\partial \gamma_i} + \frac{1}{\sigma^2}\mathbf{y}'\frac{\partial \mathbf{P}}{\partial \gamma_i}\mathbf{y}\right), \qquad (4.10)$$

because $\mathbf{R}$ does not depend on $\boldsymbol{\gamma}$. For the derivative of the logarithm of the determinant of a matrix we can use Jacobi's formula stating that for an invertible matrix $\mathbf{A}$ [60]

$$\frac{\partial |\mathbf{A}|}{\partial t} = |\mathbf{A}| \operatorname{tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial t}\right)$$

and thus

$$\begin{aligned}\frac{\partial \log|\mathbf{A}|}{\partial t} &= \frac{1}{|\mathbf{A}|}\frac{\partial |\mathbf{A}|}{\partial t} \\ &= \operatorname{tr}\left(\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial t}\right).\end{aligned}$$

As such, denoting $\frac{\partial \mathbf{G}}{\partial \gamma_i}$ by $\dot{\mathbf{G}}_i$, Eq. (4.10) becomes

$$\frac{\partial l_{\mathrm{REML}}(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi})}{\partial \gamma_i} = -\left(\operatorname{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \gamma_i}\right) + \operatorname{tr}\left(\mathbf{G}^{-1}\dot{\mathbf{G}}_i\right) + \frac{1}{\sigma^2}\mathbf{y}'\frac{\partial \mathbf{P}}{\partial \gamma_i}\mathbf{y}\right). \qquad (4.11)$$

We know that the coefficient matrix $\mathbf{C}$ of the mixed model equations only depends on $\boldsymbol{\gamma}$ through the presence of $\mathbf{G}^{-1}$ in its lower right block $\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1}$. Therefore,

$$\frac{\partial \mathbf{C}}{\partial \gamma_i} = \begin{bmatrix} \mathbf{0}_{k\times k} & \mathbf{0}_{k\times l} \\ \mathbf{0}_{l\times k} & \frac{\partial \mathbf{G}^{-1}}{\partial \gamma_i} \end{bmatrix},$$

where

$$\frac{\partial \mathbf{G}^{-1}}{\partial \gamma_i} = -\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}$$

and $\mathbf{0}_{i\times j}$ is the null matrix of dimension $i \times j$. If we then write the inverse

of $\mathbf{C}$ as

$$\mathbf{C}^{-1} = \begin{bmatrix} \mathbf{C}^{XX} & \mathbf{C}^{XZ} \\ \mathbf{C}^{ZX} & \mathbf{C}^{ZZ} \end{bmatrix},$$

with $\mathbf{C}^{XX}$ of dimension $k \times k$, $\mathbf{C}^{ZZ}$ of dimension $l \times l$, $\mathbf{C}^{XZ}$ of dimension $k \times l$ and $\mathbf{C}^{ZX}$ of dimension $l \times k$, the first term of Eq. (4.11) becomes

$$\begin{aligned} \operatorname{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \gamma_i}\right) &= \operatorname{tr}\left(\begin{bmatrix} \mathbf{C}^{XX} & \mathbf{C}^{XZ} \\ \mathbf{C}^{ZX} & \mathbf{C}^{ZZ} \end{bmatrix}\begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1} \end{bmatrix}\right) \\ &= \operatorname{tr}\left(\begin{bmatrix} \mathbf{0} & \mathbf{C}^{XZ} - \mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1} \\ \mathbf{0} & -\mathbf{C}^{ZZ}\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1} \end{bmatrix}\right) \\ &= -\operatorname{tr}\left(\mathbf{C}^{ZZ}\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}\right). \end{aligned}$$

For the last term of Eq. (4.11), we will first derive a general form for the derivative of $\mathbf{P}$ with respect to any variance component κ_i on which $\mathbf{V}$ relies:

$$\begin{aligned} \frac{\partial \mathbf{P}}{\partial \kappa_i} &= \frac{\partial \mathbf{V}^{-1}}{\partial \kappa_i} - \frac{\partial}{\partial \kappa_i}\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} \\ &= -\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial \kappa_i}\mathbf{V}^{-1} + \mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial \kappa_i}\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} \\ &\quad -\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial \kappa_i}\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} \\ &\quad +\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial \kappa_i}\mathbf{V}^{-1} \\ &= -\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial \kappa_i}\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right) \\ &\quad +\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\frac{\partial \mathbf{V}}{\partial \kappa_i}\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right) \\ &= -\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right)\frac{\partial \mathbf{V}}{\partial \kappa_i}\mathbf{P} \\ &= -\mathbf{P}\frac{\partial \mathbf{V}}{\partial \kappa_i}\mathbf{P}. \end{aligned}$$

In particular for the variance components in $\boldsymbol{\gamma}$, we obtain

$$\begin{aligned} \mathbf{y}'\frac{\partial \mathbf{P}}{\partial \gamma_i}\mathbf{y} &= -\mathbf{y}'\mathbf{P}\frac{\partial \mathbf{V}}{\partial \gamma_i}\mathbf{P}\mathbf{y} \\ &= -\mathbf{y}'\mathbf{P}\mathbf{Z}\dot{\mathbf{G}}_i\mathbf{Z}'\mathbf{P}\mathbf{y} \end{aligned}$$

and from the previous chapter we have learned that the BLUP of $\mathbf{u}$ was

found by

$$\begin{aligned}\hat{\mathbf{u}} &= \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \\ &= \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\left(\mathbf{y} - \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{y}\right) \\ &= \mathbf{G}\mathbf{Z}'\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right)\mathbf{y} \\ &= \mathbf{G}\mathbf{Z}'\mathbf{P}\mathbf{y}\,,\end{aligned}$$

where we have introduced the BLUE of $\boldsymbol{\beta}$ in the second step. Using these results we can find the REML estimates of the variance components in $\boldsymbol{\gamma}$ by equating to zero

$$\begin{aligned}\frac{\partial l_{\mathrm{REML}}(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi})}{\partial \gamma_i} = \operatorname{tr}\left(\mathbf{C}^{ZZ}\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}\right) - \operatorname{tr}\left(\mathbf{G}^{-1}\dot{\mathbf{G}}_i\right) \\ + \frac{\hat{\mathbf{u}}'\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}\hat{\mathbf{u}}}{\sigma^2}\,.\end{aligned} \tag{4.12}$$

The variance components ϕ_i included in $\boldsymbol{\phi}$ can be found analogously:

$$\begin{aligned}\frac{\partial l_{\mathrm{REML}}\left(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}\right)}{\partial \phi_i} &= -\left(\frac{\partial \log|\mathbf{C}|}{\partial \phi_i} + \frac{\partial \log|\mathbf{R}|}{\partial \phi_i} + \frac{1}{\sigma^2}\mathbf{y}'\frac{\partial \mathbf{P}}{\partial \phi_i}\mathbf{y}\right) \\ &= -\left(\operatorname{tr}\left(\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial \phi_i}\right) + \operatorname{tr}\left(\mathbf{R}^{-1}\dot{\mathbf{R}}_i\right)\right. \\ &\quad \left. -\frac{1}{\sigma^2}\mathbf{y}'\mathbf{P}\frac{\partial \mathbf{V}}{\partial \phi_i}\mathbf{P}\mathbf{y}\right)\,,\end{aligned} \tag{4.13}$$

where $\frac{\partial \mathbf{R}}{\phi_i}$ is denoted by $\dot{\mathbf{R}}_i$. The first term of Eq. (4.13) contains the following expression:

$$\begin{aligned}\frac{\partial \mathbf{C}}{\partial \phi_i} &= \begin{bmatrix}\mathbf{X}'\frac{\partial \mathbf{R}^{-1}}{\partial \phi_i}\mathbf{X} & \mathbf{X}'\frac{\partial \mathbf{R}^{-1}}{\partial \phi_i}\mathbf{Z} \\ \mathbf{Z}'\frac{\partial \mathbf{R}^{-1}}{\partial \phi_i}\mathbf{X} & \mathbf{Z}'\frac{\partial \mathbf{R}^{-1}}{\partial \phi_i}\mathbf{Z}\end{bmatrix} \\ &= -\begin{bmatrix}\mathbf{X}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{Z}\end{bmatrix} \\ &= -\mathbf{W}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{W}\,,\end{aligned}$$

with $\mathbf{W} = \begin{bmatrix}\mathbf{X} & \mathbf{Z}\end{bmatrix}$. For the last term of Eq. (4.13), we make use of the fact

that $\frac{\partial \mathbf{V}}{\partial \gamma_i} = \dot{\mathbf{R}}_i$ and that

$$\begin{aligned}
\mathbf{P}\mathbf{y} = & \left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right)\mathbf{y} \\
= & \, \mathbf{V}^{-1}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \\
= & \, \left(\mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}'\right)^{-1}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \\
= & \, \left(\mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{Z}\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1}\right)^{-1}\mathbf{Z}'\mathbf{R}^{-1}\right)\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \\
= & \, \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) - \mathbf{R}^{-1}\mathbf{Z}\mathbf{G}\mathbf{Z}'\left(\mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}'\right)^{-1}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right) \\
= & \, \mathbf{R}^{-1}\left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}} - \mathbf{Z}\hat{\mathbf{u}}\right) \\
= & \, \mathbf{R}^{-1}\hat{\mathbf{e}}\,,
\end{aligned}$$

where we again used some equalities from the blockwise inversion of a matrix with 2 invertible diagonal blocks in steps 3 and 4 and where $\hat{\mathbf{e}}$ is the vector of residuals after estimation of the fixed and random effects. Eventually, the REML estimates of the variance components in $\boldsymbol{\phi}$ can be found by equating to zero

$$\begin{aligned}
\frac{\partial l_{\mathrm{REML}}\left(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}\right)}{\partial \phi_i} = & \operatorname{tr}\left(\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{W}\right) - \operatorname{tr}\left(\mathbf{R}^{-1}\dot{\mathbf{R}}_i\right) \\
& + \frac{\hat{\mathbf{e}}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\hat{\mathbf{e}}}{\sigma^2}\,. \qquad (4.14)
\end{aligned}$$

It can be seen that an analytical solution for the REML estimates of $\boldsymbol{\phi}$ and $\boldsymbol{\gamma}$ is almost never attainable even for simple covariance structures due to the presence of (a part of) the inverse of $\mathbf{C}$ in the first term of Eq. (4.12) and (4.14). Therefore, we have to resort to iterative techniques for finding the REML estimates of $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$. A well-known iterative technique for maximising a likelihood is the Newton-Raphson method which relies on the first derivatives with respect to the parameters to be estimated of the likelihood and the Hessian matrix. The Hessian matrix contains the second derivatives with respect to the parameters to be estimated of the likelihood, but may be tedious to construct. Therefore, it can be replaced by its expected value, the so-called Fisher information matrix, which was also used by Patterson and Thompson, because it was easier to construct compared to the Hessian matrix [59]. Other quasi-Newton methods make use of other approximations of the Hessian matrix when it is too difficult to

construct the Hessian or Fisher information matrix or if they are non-existent. Another famous and widely-used method for maximising the REML log-likelihood is the expectation-maximisation (EM) algorithm, which does not need the calculation of a (quasi-)Hessian matrix [64]. However, although the computing time per iteration decreases compared to quasi-Newton methods, the number of iterations needed to converge is usually a lot higher resulting in a higher overall computation time.

In this dissertation, we focus on a variant of the Newton-Raphson method called the Average Information algorithm, as it was shown by Gilmour et al., in a plant breeding context, and Johnson and Thompson, in an animal breeding context, that this method is a viable alternative when problems get larger in number of observations and number of variance components [65, 66]. In the following sections we will give more details about the average information algorithm and explain why it has been chosen in this dissertation as the method for maximising the REML log-likelihood, compared to some other alternatives.

4.3. Average Information Algorithm

The average information algorithm for maximising the REML log-likelihood is mostly referred to as the AI-REML algorithm. It is a quasi-Newton method as it uses a simplified average of the Hessian and the Fisher information matrix to obtain updates for the REML estimates at each iteration. The original Newton-Raphson method for maximising a function f with respect to a set of parameters θ is the following:

$$\boldsymbol{\theta}^{(m+1)} = \boldsymbol{\theta}^{(m)} - \left(\mathbf{H}^{(m)}\right)^{-1} \nabla f\left(\theta^{(m)}\right), \tag{4.15}$$

where $\boldsymbol{\theta}^{(m)}$ is the estimated value for the vector of parameters at iteration m, $\mathbf{H}^{(m)}$ is the Hessian matrix with the second derivatives of the function f with respect to $\boldsymbol{\theta}$ evaluated for $\boldsymbol{\theta}^{(m)}$ and $\nabla f\left(\theta^{(m)}\right)$ is the gradient vector of f with respect to $\boldsymbol{\theta}$ evaluated for $\boldsymbol{\theta}^{(m)}$. In our case, for the maximisation of

the REML log-likelihood , $\nabla f\left(\theta\right)$ takes the form

$$\nabla l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi}\right)=\begin{bmatrix}\frac{\partial l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\sigma^2}\\ \frac{\partial l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\boldsymbol{\gamma}}\\ \frac{\partial l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\boldsymbol{\phi}}\end{bmatrix}=\begin{bmatrix}\frac{\partial}{\partial\sigma^2}\\ \frac{\partial}{\partial\gamma_1}\\ \vdots\\ \frac{\partial}{\partial\gamma_p}\\ \frac{\partial}{\partial\phi_1}\\ \vdots\\ \frac{\partial}{\partial\phi_q}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,,$$

with p the number of variance components in vector $\boldsymbol{\gamma}$ and q the number of variance components included in vector $\boldsymbol{\phi}$. The elements of this vector can be calculated using the results in Eq. (4.9), (4.12) and (4.14).

The Hessian matrix has the following form in our case

$$\mathbf{H}=\begin{bmatrix}\frac{\partial^2}{(\partial\sigma^2)^2} & \frac{\partial^2}{\partial\sigma^2\partial\boldsymbol{\gamma}'} & \frac{\partial^2}{\partial\sigma^2\partial\boldsymbol{\phi}'}\\ \frac{\partial^2}{\partial\sigma^2\partial\boldsymbol{\gamma}} & \frac{\partial^2}{\partial\boldsymbol{\gamma}\partial\boldsymbol{\gamma}'} & \frac{\partial^2}{\partial\boldsymbol{\gamma}\partial\boldsymbol{\phi}'}\\ \frac{\partial^2}{\partial\sigma^2\partial\boldsymbol{\phi}} & \frac{\partial^2}{\partial\boldsymbol{\phi}\partial\boldsymbol{\gamma}'} & \frac{\partial^2}{\partial\boldsymbol{\phi}\partial\boldsymbol{\phi}'}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,,$$

which is symmetric and contains the following elements:

$$\frac{\partial^2 l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\sigma^2\partial\boldsymbol{\gamma}'}=\begin{bmatrix}\frac{\partial^2}{\partial\sigma^2\partial\gamma_1} & \cdots & \frac{\partial^2}{\partial\sigma^2\partial\gamma_p}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,,$$

$$\frac{\partial^2 l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\sigma^2\partial\boldsymbol{\phi}'}=\begin{bmatrix}\frac{\partial^2}{\partial\sigma^2\partial\phi_1} & \cdots & \frac{\partial^2}{\partial\sigma^2\partial\phi_q}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,,$$

$$\frac{\partial^2 l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\boldsymbol{\gamma}\partial\boldsymbol{\gamma}'}=\begin{bmatrix}\frac{\partial^2}{\partial\gamma_1\partial\gamma_1} & \cdots & \frac{\partial^2}{\partial\gamma_1\partial\gamma_p}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2}{\partial\gamma_p\partial\gamma_1} & \cdots & \frac{\partial^2}{\partial\gamma_p\partial\gamma_p}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,,$$

$$\frac{\partial^2 l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\boldsymbol{\gamma}\partial\boldsymbol{\phi}'}=\begin{bmatrix}\frac{\partial^2}{\partial\gamma_1\partial\phi_1} & \cdots & \frac{\partial^2}{\partial\gamma_1\partial\phi_q}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2}{\partial\gamma_p\partial\phi_1} & \cdots & \frac{\partial^2}{\partial\gamma_p\partial\phi_q}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,,$$

$$\frac{\partial^2 l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\boldsymbol{\phi}\partial\boldsymbol{\phi}'}=\begin{bmatrix}\frac{\partial^2}{\partial\phi_1\partial\phi_1} & \cdots & \frac{\partial^2}{\partial\phi_1\partial\phi_q}\\ \vdots & \ddots & \vdots\\ \frac{\partial^2}{\partial\phi_p\partial\phi_1} & \cdots & \frac{\partial^2}{\partial\phi_p\partial\phi_q}\end{bmatrix}l_{\mathrm{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})\,.$$

However, taking the second derivatives starting from the first derivatives in Eq. (4.12) and (4.14) is not so convenient. Therefore, a more general form of the first derivative of the REML log-likelihood with respect to any variance component κ_i on which $\mathbf{V}$ relies is derived, based on the expression in Eq. (4.7):

$$\begin{aligned}
\frac{\partial l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\kappa_i} &= -\left(\frac{\partial\log\left|\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right|}{\partial\kappa_i}+\frac{\partial\log|\mathbf{V}|}{\partial\kappa_i}+\frac{1}{\sigma^2}\mathbf{y}'\frac{\mathbf{P}}{\partial\kappa_i}\mathbf{y}\right)\\
&= \mathrm{tr}\left(\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\dot{\mathbf{V}}_i\mathbf{V}^{-1}\mathbf{X}\right)-\mathrm{tr}\left(\mathbf{V}^{-1}\dot{\mathbf{V}}_i\right)\\
&\quad+\frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^2}\\
&= \frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^2}-\mathrm{tr}\left(\mathbf{V}^{-1}\dot{\mathbf{V}}_i\right)\\
&\quad+\mathrm{tr}\left(\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\dot{\mathbf{V}}_i\right)\\
&= \frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^2}\\
&\quad-\mathrm{tr}\left(\left(\mathbf{V}^{-1}-\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right)\dot{\mathbf{V}}_i\right)\\
&= \frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^2}-\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right), \qquad (4.16)
\end{aligned}$$

where κ_i are the elements of the vector $\boldsymbol{\kappa}=(\boldsymbol{\gamma}',\boldsymbol{\phi}')'$ and $\dot{\mathbf{V}}_i=\frac{\partial\mathbf{V}}{\partial\kappa_i}$. The second derivatives can then be found based on Eq. (4.9) and (4.16)

$$\begin{aligned}
\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\left(\partial\sigma^2\right)^2} &= \frac{n-k}{\sigma^4}-\frac{2\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^6},\\
\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\sigma^2\partial\kappa_i} &= -\frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^4},\\
\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\kappa_i\partial\kappa_j} &= \frac{1}{\sigma^2}\mathbf{y}'\frac{\partial\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}}{\partial\kappa_j}\mathbf{y}-\frac{\partial\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right)}{\partial\kappa_j}. \qquad (4.17)
\end{aligned}$$

The last equation in (4.17) needs some more work to be evaluated. Firstly, the trace function is a linear operator on the matrix elements and thus it

commutes with the derivative, implying that

$$\begin{aligned}\frac{\partial \mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right)}{\partial \kappa_j} &= \mathrm{tr}\left(\frac{\partial \mathbf{P}\dot{\mathbf{V}}_i}{\partial \kappa_j}\right) \\ &= \mathrm{tr}\left(\mathbf{P}\frac{\partial \dot{\mathbf{V}}_i}{\partial \kappa_j} + \frac{\partial \mathbf{P}}{\partial \kappa_j}\dot{\mathbf{V}}_i\right) \\ &= \mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) - \mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\dot{\mathbf{V}}_i\right)\,,\end{aligned}$$

where we have denoted $\frac{\partial^2 \mathbf{V}}{\partial \kappa_i \partial \kappa_j}$ by $\ddot{\mathbf{V}}_{ij}$. For the evaluation of $\frac{\partial \mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}}{\partial \kappa_j}$, we use the well-known product rule for derivation:

$$\begin{aligned}\mathbf{y}'\frac{\partial \mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}}{\partial \kappa_j}\mathbf{y} &= \mathbf{y}'\left(\frac{\partial \mathbf{P}}{\partial \kappa_j}\dot{\mathbf{V}}_i\mathbf{P} + \mathbf{P}\frac{\partial \dot{\mathbf{V}}_i}{\partial \kappa_j}\mathbf{P} + \mathbf{P}\dot{\mathbf{V}}_i\frac{\partial \mathbf{P}}{\partial \kappa_j}\right)\mathbf{y} \\ &= \mathbf{y}'\left(-\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P} + \mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P} - \mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\right)\mathbf{y} \\ &= \mathbf{y}'\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{y} - 2\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{y}\,,\end{aligned}$$

where the last step holds because $\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{y}$ is a scalar and thus equal to its transpose $\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}$. As such the second derivative with respect to elements of $\boldsymbol{\kappa}$ can be written as

$$\begin{aligned}\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2, \boldsymbol{\kappa}\right)}{\partial \kappa_i \partial \kappa_j} = \mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\dot{\mathbf{V}}_i\right) - \mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) + \frac{\mathbf{y}'\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{y}}{\sigma^2} \\ - \frac{2\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{y}}{\sigma^2}\,.\end{aligned}$$

This expression can be hard to evaluate because traces are needed of some matrices that are tedious to calculate. Another option is to use the Fisher information matrix, which is the expected value of the Hessian matrix, instead of the observed Hessian matrix as used originally in Eq. (4.15) for updating the variance components. The expected Hessian can be derived from the observed Hessian using the following theorem for quadratic forms for any symmetric matrix $\mathbf{A}$ and a random variable $\mathbf{y}$ with $\mathrm{E}(\mathbf{y}) = \boldsymbol{\mu}$ and $\mathrm{var}(\mathbf{y}) = \mathbf{Q}$ [67]:

$$\mathrm{E}\left(\mathbf{y}'\mathbf{A}\mathbf{y}\right) = \mathrm{tr}\left(\mathbf{A}\mathbf{Q}\right) + \boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}\,.$$

This theorem can be extended for non-symmetric matrices $\mathbf{B}$ by noticing

that $\mathbf{y}'\mathbf{B}'\mathbf{y}$ is equal to $\mathbf{y}'\mathbf{B}\mathbf{y}$ and thus

$$\mathrm{E}\left(\mathbf{y}'\mathbf{B}\mathbf{y}\right) = \frac{1}{2}\left(\mathrm{E}\left(\mathbf{y}'\mathbf{B}\mathbf{y}\right) + \mathrm{E}\left(\mathbf{y}'\mathbf{B}'\mathbf{y}\right)\right) = \frac{1}{2}\mathrm{E}\left(\mathbf{y}'\left(\mathbf{B}+\mathbf{B}'\right)\mathbf{y}\right) ,$$

which is again a quadratic form with a symmetric matrix $\mathbf{B}+\mathbf{B}'$ and so

$$\begin{aligned}
\mathrm{E}\left(\mathbf{y}'\mathbf{B}\mathbf{y}\right) &= \frac{1}{2}\mathrm{tr}\left(\left(\mathbf{B}+\mathbf{B}'\right)\mathbf{Q}\right) + \frac{1}{2}\boldsymbol{\mu}'\left(\mathbf{B}+\mathbf{B}'\right)\boldsymbol{\mu} \\
&= \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{B}\mathbf{Q}\right) + \mathrm{tr}\left(\mathbf{B}'\mathbf{Q}\right) + \boldsymbol{\mu}'\mathbf{B}\boldsymbol{\mu} + \boldsymbol{\mu}'\mathbf{B}'\boldsymbol{\mu}\right) \\
&= \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{Q}\mathbf{B}\right) + \mathrm{tr}\left(\mathbf{Q}\mathbf{B}\right) + 2\boldsymbol{\mu}'\mathbf{B}\boldsymbol{\mu}\right) \\
&= \mathrm{tr}\left(\mathbf{B}\mathbf{Q}\right) + \boldsymbol{\mu}'\mathbf{B}\boldsymbol{\mu} ,
\end{aligned}$$

where we have made use of the cyclic property of the trace operator $\mathrm{tr}(\mathbf{AB}) = \mathrm{tr}(\mathbf{BA})$ and the fact that the trace of a matrix and the trace of its transpose are equal: $\mathrm{tr}(\mathbf{A}) = \mathrm{tr}(\mathbf{A}')$. Another important property of $\mathbf{P}$ will also be used, namely

$$\begin{aligned}
\mathbf{PVP} &= \left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right)\mathbf{V}\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right) \\
&= \left(\mathbf{I}_n - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\right)\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right) \\
&= \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} \\
&\quad + \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} \\
&= \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1} \\
&= \mathbf{P} .
\end{aligned}$$

Let us now calculate the expected values of the different terms that are present in the observed Hessian matrix:

$$\begin{aligned}
\mathrm{E}\left(\mathbf{y}'\mathbf{P}\mathbf{y}\right) &= \sigma^2\mathrm{tr}\left(\mathbf{P}\mathbf{V}\right) + \boldsymbol{\beta}'\mathbf{X}'\mathbf{P}\mathbf{X}\boldsymbol{\beta} \\
&= \sigma^2\mathrm{tr}\left(\mathbf{I}_n - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\right) \\
&= \sigma^2\left(\mathrm{tr}\left(\mathbf{I}_n\right) - \mathrm{tr}\left(\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\right)\right) \\
&= \sigma^2\left(n - \mathrm{tr}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\right)\right) \\
&= \sigma^2\left(n - \mathrm{tr}\left(\mathbf{I}_k\right)\right) \\
&= \sigma^2\left(n-k\right) ,
\end{aligned}$$

$$
\begin{aligned}
\mathrm{E}\left(\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}\right) &= \sigma^2\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{V}\right) + \boldsymbol{\beta}'\mathbf{X}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{X}\boldsymbol{\beta} \\
&= \sigma^2\mathrm{tr}\left(\mathbf{P}\mathbf{V}\mathbf{P}\dot{\mathbf{V}}_i\right) \\
&= \sigma^2\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right) , \\
\mathrm{E}\left(\mathbf{y}'\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{y}\right) &= \sigma^2\mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{V}\right) + \boldsymbol{\beta}'\mathbf{X}'\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{X}\boldsymbol{\beta} \\
&= \sigma^2\mathrm{tr}\left(\mathbf{P}\mathbf{V}\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) \\
&= \sigma^2\mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) , \\
\mathrm{E}\left(\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{y}\right) &= \sigma^2\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{V}\right) + \boldsymbol{\beta}'\mathbf{X}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{X}\boldsymbol{\beta} \\
&= \sigma^2\mathrm{tr}\left(\mathbf{P}\mathbf{V}\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\right) \\
&= \sigma^2\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\right) ,
\end{aligned}
$$

where we have made use of the fact that $\mathbf{PX} = \mathbf{0}$ and of the cyclic property of traces. Finally the elements of the Fisher information matrix can be calculated as

$$
\begin{aligned}
\mathrm{E}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2, \boldsymbol{\kappa}\right)}{\left(\partial\sigma^2\right)^2}\right) &= \frac{n-k}{\sigma^4} - \frac{2\sigma^2\left(n-k\right)}{\sigma^6} = -\frac{n-k}{\sigma^4} , \\
\mathrm{E}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2, \boldsymbol{\kappa}\right)}{\partial\sigma^2\partial\kappa_i}\right) &= -\frac{\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right)}{\sigma^2} , \\
\mathrm{E}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2, \boldsymbol{\kappa}\right)}{\partial\kappa_i\partial\kappa_j}\right) &= \mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\dot{\mathbf{V}}_i\right) - \mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) + \mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) \\
&\quad - 2\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\right) \\
&= -\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\right) .
\end{aligned}
$$

These still require the evaluation of the traces of matrices that are tedious to construct and thus the average information algorithm uses a simplified average of both the observed and the expected Hessian matrix, which is

called the average information matrix and shall be denoted by $\mathcal{I}_A$:

$$\begin{aligned}\mathcal{I}_A &= \begin{bmatrix} \mathcal{I}_A\left(\sigma^2,\sigma^2\right) & \mathcal{I}_A\left(\sigma^2,\boldsymbol{\kappa}\right) \\ \mathcal{I}_A\left(\sigma^2,\boldsymbol{\kappa}\right) & \mathcal{I}_A\left(\boldsymbol{\kappa},\boldsymbol{\kappa}\right) \end{bmatrix} \\ &= \begin{bmatrix} \mathcal{I}_A\left(\sigma^2,\sigma^2\right) & \mathcal{I}_A\left(\sigma^2,\kappa_1\right) & \cdots & \mathcal{I}_A\left(\sigma^2,\kappa_r\right) \\ \mathcal{I}_A\left(\sigma^2,\kappa_1\right) & \mathcal{I}_A\left(\kappa_1,\kappa_1\right) & \cdots & \mathcal{I}_A\left(\kappa_1,\kappa_r\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{I}_A\left(\sigma^2,\kappa_r\right) & \mathcal{I}_A\left(\kappa_1,\kappa_r\right) & \cdots & \mathcal{I}_A\left(\kappa_r,\kappa_r\right) \end{bmatrix},\end{aligned}$$

with $r = p + q$ the total number of variance components on which $\mathbf{V}$ relies. The different components of $\mathcal{I}_A$ can be calculated as

$$\begin{aligned}\mathcal{I}_A\left(\sigma^2,\sigma^2\right) &= \frac{1}{2}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\left(\partial\sigma^2\right)^2} + \mathrm{E}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\left(\partial\sigma^2\right)^2}\right)\right) \\ &= \frac{1}{2}\left(\frac{n-k}{\sigma^4} - \frac{2\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^6} - \frac{n-k}{\sigma^4}\right) \\ &= \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^6}, \\ \mathcal{I}_A\left(\sigma^2,\kappa_i\right) &= \frac{1}{2}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\sigma^2\partial\kappa_i} + \mathrm{E}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\sigma^2\partial\kappa_i}\right)\right) \\ &= \frac{1}{2}\left(-\frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^4} - \frac{\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right)}{\sigma^2}\right) \\ &= -\frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}}{\sigma^4}, \\ \mathcal{I}_A\left(\kappa_i,\kappa_j\right) &= \frac{1}{2}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\kappa_i\partial\kappa_j} + \mathrm{E}\left(\frac{\partial^2 l_{\mathrm{REML}}\left(\sigma^2,\boldsymbol{\kappa}\right)}{\partial\kappa_i\partial\kappa_j}\right)\right) \\ &= \frac{1}{2}\left(\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\dot{\mathbf{V}}_i\right) - \mathrm{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right) + \frac{\mathbf{y}'\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{y}}{\sigma^2}\right. \\ &\quad \left. -\frac{2\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{y}}{\sigma^2} - \mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\right)\right) \\ &= \frac{\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\dot{\mathbf{V}}_j\mathbf{P}\mathbf{y}}{\sigma^2},\end{aligned}$$

where $\mathrm{tr}\left(\mathbf{P}\dot{\mathbf{V}}_i\right)$ is approximated by $\frac{1}{\sigma^2}\mathbf{y}'\mathbf{P}\dot{\mathbf{V}}_i\mathbf{P}\mathbf{y}$ because maximising the likelihood means that Eq. (4.16) approaches zero and where $\mathbf{y}'\mathbf{P}\ddot{\mathbf{V}}_{ij}\mathbf{P}\mathbf{y}$ is

approximated by its expected value $\sigma^2 \text{tr}\left(\mathbf{P}\ddot{\mathbf{V}}_{ij}\right)$. Note that these last terms also disappear whenever $\mathbf{V}$ is a linear function of $\boldsymbol{\kappa}$. As such, the elements of $\mathcal{I}_A$ do no longer involve the calculation of traces of matrices and are, therefore, much easier to calculate than the elements of the Hessian or Fisher information matrix.

4.4. Computational aspects

4.4.1. Computational convenience of the AI-REML algorithm

Although we mentioned that the computation of the average information matrix is no longer as hard as calculating the Hessian or Fisher information matrix, it still seems a computational burden since we might need to calculate $\mathbf{P}$. However, this can be avoided by recognizing that the average information matrix can be written as:

$$\mathcal{I}_A = \frac{1}{\sigma^2}\mathbf{Q}'\mathbf{P}\mathbf{Q}\,,$$

with

$$\mathbf{Q} = \left[\tfrac{1}{\sigma^2}\mathbf{y} \quad \dot{\mathbf{V}}_1\mathbf{P}\mathbf{y} \quad \cdots \quad \dot{\mathbf{V}}_r\mathbf{P}\mathbf{y}\right].$$

If we again look at the specific variance components $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$, the matrix $\mathbf{Q}$ can be written as

$$\mathbf{Q} = \left[\tfrac{1}{\sigma^2}\mathbf{y} \quad \tfrac{\partial \mathbf{V}}{\partial \gamma_1}\mathbf{P}\mathbf{y} \quad \cdots \quad \tfrac{\partial \mathbf{V}}{\partial \gamma_p}\mathbf{P}\mathbf{y} \quad \tfrac{\partial \mathbf{V}}{\partial \phi_1}\mathbf{P}\mathbf{y} \quad \cdots \quad \tfrac{\partial \mathbf{V}}{\partial \phi_q}\mathbf{P}\mathbf{y}\right],$$

with

$$\frac{\partial \mathbf{V}}{\partial \gamma_i}\mathbf{P}\mathbf{y} = \mathbf{Z}\dot{\mathbf{G}}_i\mathbf{Z}'\mathbf{P}\mathbf{y} = \mathbf{Z}\dot{\mathbf{G}}_i\mathbf{G}^{-1}\hat{\mathbf{u}}\,,$$
$$\frac{\partial \mathbf{V}}{\partial \phi_i}\mathbf{P}\mathbf{y} = \dot{\mathbf{R}}_i\mathbf{P}\mathbf{y} = \dot{\mathbf{R}}_i\mathbf{R}^{-1}\hat{\mathbf{e}}\,.$$

To obtain $\mathbf{Q}'\mathbf{P}\mathbf{Q}$ in an easy way, we will first construct an augmented form of the coefficient matrix $\mathbf{C}$:

$$\begin{aligned}\mathbf{M} &= \begin{bmatrix} \mathbf{Q}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{Q}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Q}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{X}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Q}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{Q}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{C} \end{bmatrix}\end{aligned}$$

with $\mathbf{W} = \begin{bmatrix}\mathbf{X} & \mathbf{Z}\end{bmatrix}$ as defined before. The Schur complement of the coefficient matrix $\mathbf{C}$ in this matrix is defined as

$$\begin{aligned}\mathbf{M}/\mathbf{C} &= \mathbf{Q}'\mathbf{R}^{-1}\mathbf{Q} - \mathbf{Q}'\mathbf{R}^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}\mathbf{Q} \\ &= \mathbf{Q}'\left(\mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}\right)\mathbf{Q}\,. \end{aligned} \tag{4.18}$$

If we define the inverse of $\mathbf{C}$ as

$$\mathbf{C}^{-1} = \begin{bmatrix} \mathbf{C}^{XX} & \mathbf{C}^{XZ} \\ \mathbf{C}^{ZX} & \mathbf{C}^{ZZ} \end{bmatrix},$$

then we can obtain an expression for each part of the inverse using the blockwise inversion of a matrix:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & -\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix},$$

and so

$$\begin{aligned}\mathbf{C}^{XX} &= \left(\mathbf{X}'\mathbf{R}^{-1}\mathbf{X} - \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1}\right)^{-1}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}\right)^{-1} \\ &= \left(\mathbf{X}'\left(\mathbf{R}^{-1} - \mathbf{R}^{-1}\mathbf{Z}\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1}\right)^{-1}\mathbf{Z}'\mathbf{R}^{-1}\right)\mathbf{X}\right)^{-1} \\ &= \left(\mathbf{X}'\left(\mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}'\right)^{-1}\mathbf{X}\right)^{-1} \\ &= \left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1},\end{aligned}$$

$$
\begin{aligned}
\mathbf{C}^{ZX} &= -\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right)^{-1}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}\mathbf{C}^{XX} \\
&= -\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right)^{-1}\mathbf{Z}'\mathbf{R}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1} \\
&= -\mathbf{G}\mathbf{Z}'\left(\mathbf{R}+\mathbf{Z}\mathbf{G}\mathbf{Z}'\right)^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1} \\
&= -\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1} = \left(\mathbf{C}^{XZ}\right)', \\
\mathbf{C}^{ZZ} &= \left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right)^{-1} - \mathbf{C}^{ZX}\mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right)^{-1} \\
&= \left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right)^{-1} \\
&\quad + \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}\left(\mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z}+\mathbf{G}^{-1}\right)^{-1} \\
&= \mathbf{G} - \mathbf{G}\mathbf{Z}'\left(\mathbf{R}+\mathbf{Z}\mathbf{G}\mathbf{Z}'\right)^{-1}\mathbf{Z}\mathbf{G} \\
&\quad + \mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{Z}\mathbf{G} \\
&= \mathbf{G} - \mathbf{G}\mathbf{Z}'\left(\mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\right)\mathbf{Z}\mathbf{G} \\
&= \mathbf{G} - \mathbf{G}\mathbf{Z}'\mathbf{P}\mathbf{Z}\mathbf{G},
\end{aligned}
$$

where $\mathbf{C}^{ZX} = \left(\mathbf{C}^{XZ}\right)'$, because $\mathbf{C}$ is a symmetric matrix and thus its inverse is also symmetric. Using these expressions, we can derive a convenient form of $\mathbf{W}\mathbf{C}^{-1}\mathbf{W}$:

$$
\begin{aligned}
\mathbf{W}\mathbf{C}^{-1}\mathbf{W} &= \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' - \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{Z}\mathbf{G}\mathbf{Z}' \\
&\quad - \mathbf{Z}\mathbf{G}\mathbf{Z}'\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' + \mathbf{Z}\mathbf{G}\mathbf{Z}' - \mathbf{Z}\mathbf{G}\mathbf{Z}'\mathbf{P}\mathbf{Z}\mathbf{G}\mathbf{Z}' \\
&= \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' - \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' \\
&\quad + \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{R} - \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' \\
&\quad + \mathbf{R}\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' + \mathbf{V} - \mathbf{R} - \mathbf{V}\mathbf{P}\mathbf{V} + \mathbf{V}\mathbf{P}\mathbf{R} \\
&\quad + \mathbf{R}\mathbf{P}\mathbf{V} - \mathbf{R}\mathbf{P}\mathbf{R} \\
&= \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{R} - \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' \\
&\quad + \mathbf{R}\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' + \mathbf{V} - \mathbf{R} - \mathbf{V} \\
&\quad + \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' + \mathbf{R} - \mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}\mathbf{R} + \mathbf{R} \\
&\quad - \mathbf{R}\mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}' - \mathbf{R}\mathbf{P}\mathbf{R} \\
&= \mathbf{R} - \mathbf{R}\mathbf{P}\mathbf{R},
\end{aligned}
$$

where in the first step we used the fact that $\mathbf{V} = \mathbf{Z}\mathbf{G}\mathbf{Z} + \mathbf{R}$ and thus $\mathbf{Z}\mathbf{G}\mathbf{Z} = \mathbf{V} - \mathbf{R}$. Therefore, the Schur complement of $\mathbf{C}$ in matrix $\mathbf{M}$ is equal

to:

$$\begin{aligned}\mathbf{Q}'\left(\mathbf{R}^{-1}-\mathbf{R}^{-1}\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}\right)\mathbf{Q} &= \mathbf{Q}'\left(\mathbf{R}^{-1}-\mathbf{R}^{-1}\left(\mathbf{R}-\mathbf{R}\mathbf{P}\mathbf{R}\right)\mathbf{R}^{-1}\right)\mathbf{Q}\\ &= \mathbf{Q}'\left(\mathbf{R}^{-1}-\mathbf{R}^{-1}+\mathbf{P}\right)\mathbf{Q}\\ &= \mathbf{Q}'\mathbf{P}\mathbf{Q}\,.\end{aligned}$$

Note that we did not make use of any property of matrix $\mathbf{Q}$ and so in this expression $\mathbf{Q}$ can be replaced by any other matrix or vector. As such, it is shown here that matrix $\mathbf{P}$ does not need to be calculated explicitly to obtain the average information matrix.

The whole variance component estimation process thus needs the following components:

- An initial estimate of the variance components.
- The first derivatives of the REML log-likelihood with respect to the different variance components, also called the score functions:

$$\frac{\partial l_{\text{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\sigma^2} = -\left(\frac{n-k}{\sigma^2}-\frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^4}\right), \tag{4.19}$$

$$\begin{aligned}\frac{\partial l_{\text{REML}}(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi})}{\partial\gamma_i} &= \operatorname{tr}\left(\mathbf{C}^{ZZ}\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}\right)-\operatorname{tr}\left(\mathbf{G}^{-1}\dot{\mathbf{G}}_i\right)\\ &\quad+\frac{\hat{\mathbf{u}}'\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}\hat{\mathbf{u}}}{\sigma^2}\,,\end{aligned} \tag{4.20}$$

$$\begin{aligned}\frac{\partial l_{\text{REML}}\left(\sigma^2,\boldsymbol{\gamma},\boldsymbol{\phi}\right)}{\partial\phi_i} &= \operatorname{tr}\left(\mathbf{C}^{-1}\mathbf{W}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\mathbf{W}\right)-\operatorname{tr}\left(\mathbf{R}^{-1}\dot{\mathbf{R}}_i\right)\\ &\quad+\frac{\hat{\mathbf{e}}'\mathbf{R}^{-1}\dot{\mathbf{R}}_i\mathbf{R}^{-1}\hat{\mathbf{e}}}{\sigma^2}\,.\end{aligned}$$

- The average information matrix $\mathcal{I}_A=\frac{1}{\sigma^2}\mathbf{Q}'\mathbf{P}\mathbf{Q}$, which can be found as the Schur complement of $\mathbf{C}$ in $\mathbf{M}$:

$$\mathbf{M}=\begin{bmatrix}\mathbf{Q}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{Q}'\mathbf{R}^{-1}\mathbf{W}\\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{Q} & \mathbf{C}\end{bmatrix}$$

with

$$\mathbf{Q}=\begin{bmatrix}\frac{1}{\sigma^2}\mathbf{y} & \mathbf{Z}\dot{\mathbf{G}}_1\mathbf{G}^{-1}\hat{\mathbf{u}} & \cdots & \mathbf{Z}\dot{\mathbf{G}}_p\mathbf{G}^{-1}\hat{\mathbf{u}} & \dot{\mathbf{R}}_1\mathbf{R}^{-1}\hat{\mathbf{e}} & \cdots & \dot{\mathbf{R}}_q\mathbf{R}^{-1}\hat{\mathbf{e}}\end{bmatrix}$$

Note that when $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$ have been assigned a value, there exists an analytical solution for $\hat{\sigma}^2 = \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{n-k}$ by equating the first score function in Eq. (4.19) to zero. First of all, we see that for evaluating the score functions and setting up the AI matrix, the BLUP for $\mathbf{u}$ is needed together with a prediction of $\mathbf{e}$, which also requires the BLUE of $\boldsymbol{\beta}$. The Mixed Model Equations (MME) provide an easy way of finding these estimates and predictions. One way of solving these MME is using a direct solving routine, which is based on the Cholesky decomposition of the coefficient matrix $\mathbf{C}$. The availability of the Cholesky decomposition facilitates solving other equations with $\mathbf{C}$ as coefficient matrix, because calculating the Cholesky decomposition is a task of complexity $O(m^3)$, with m the dimension of the square matrix, while solving an equation when the decomposition of the coefficient matrix is known only has complexity $O(m^2)$. So when the Cholesky decomposition of $\mathbf{C}$ is already available due to the direct solving of the MME, the AI matrix can be calculated using Eq. (4.18):

- Construct matrix $\mathbf{W}'\mathbf{R}^{-1}\mathbf{Q}$.
- Solve the equation $\mathbf{C}\mathbf{Y} = \mathbf{W}'\mathbf{R}^{-1}\mathbf{Q}$ for $\mathbf{Y}$ by using the previously obtained Cholesky decomposition of $\mathbf{C}$.
- Compute $\mathbf{Q}'\mathbf{P}\mathbf{Q} = \mathbf{Q}'\mathbf{R}^{-1}\mathbf{Q} - \mathbf{Q}'\mathbf{R}^{-1}\mathbf{W}\mathbf{Y}$.
- $\mathcal{I}_A = \frac{1}{\sigma^2}\mathbf{Q}'\mathbf{P}\mathbf{Q}$.

Moreover, for evaluating the score functions we also need the inverse of the coefficient matrix, which can also more easily be calculated by using the already obtained Cholesky decomposition of $\mathbf{C}$. In fact, the inversion of $\mathbf{C}$ then comes down to inverting its factor, which can be done by applying m forward substitions, and multiplying it with its transpose:

$$\begin{aligned}\mathbf{C} &= \mathbf{L}\mathbf{L}' \\ \mathbf{C}^{-1} &= \left(\mathbf{L}^{-1}\right)'\mathbf{L}^{-1}\end{aligned}$$

4.4.2. Comparison of AI-REML with other iterative procedures

It was already mentioned that when the REML equations were conceived in 1971 by Patterson and Thompson, the Fisher information matrix was used for iteratively searching for the values of the variance components

that maximise the likelihood [59]. However, it was also shown in the same paper that the calculation of the elements of the Fisher information matrix can become very tedious when the number of observations gets large as we need to calculate traces of different $n \times n$ matrices for each element. The expectation-maximisation algorithm, first described in 1977, solved this problem by no longer needing a Hessian or Fisher information matrix for finding an update of the variance component estimates [64]. Details of this algorithm are left for the interested reader and can be found in many textbooks and articles [43, 64]. In short, this algorithm comes down to equating the score functions to zero and solving them for the variance components, by replacing the variance components with the value of the previous iteration in $\mathbf{C}^{-1}$ and by using the values for $\hat{\mathbf{u}}$ and $\hat{\mathbf{e}}$ based on a solution of the MME with the values for the variance components of the previous iteration. The computational burden was thus decreased compared to the originally used method, however, the score functions still needed to be evaluated, implying inversion of the coefficient matrix.

The derivative-free method for maximising the REML log-likelihood, put forward in 1986, was very promising as it no longer required the inversion of the coefficient matrix due to the introduction of some approximations of the traces required in the score function [68]. Unfortunately, the decreased computing time of the derivative-free method came with the price of much poorer numerical properties and some computational optimisations of the EM algorithm closed the performance gap between EM and the derivative-free approach [69]. This was mainly possible because in the pre-sequencing period, the coefficient matrix was very sparse and thus specialised algorithms for sparse matrix inversion could be used to speed up the evaluation of the score functions (these specialised algorithms will be discussed in the next chapter). Using these optimisations, the EM algorithm was only 3 times slower per iteration compared to the derivative-free approach, but the number of iterations needed to converge was usually a lot less for EM. As such, the EM algorithm regained its value, as it also was independent of the choice of the starting value for the variance component estimations.

Still, computing time remained an issue for the EM algorithm as it usually converges in a lot more iterations than Newton-Raphson based methods, but was widely used due to the low computing time per iteration [43]. From its conception in 1995, the average information algorithm was suggested as a viable alternative to the EM algorithm as the computational burden

was greatly reduced compared to the original Fisher scoring method, while keeping the number of iterations needed for convergence low [65]. In the paper by Gilmour et al., where the AI algorithm was introduced, first tests showed already that the AI algorithm outperformed both the EM and derivative-free algorithm [65]. The only downside of the AI algorithm is the fact that it is less robust with respect to the choice of the starting values compared to the EM algorithm and thus both algorithms can still be applied next to each other, depending on the problems encountered [70]. Usually, AI-REML is proposed as the method of choice, but if AI-REML estimates depend heavily on the chosen starting values, EM might be a good alternative for finding more robust REML estimates.

After the AI algorithm, no viable alternatives were presented for maximising the REML log-likelihood and because the computational burden of both EM and AI are equivalent, they struggle with the same computational limitations. A major difficulty was introduced by the application of genetic markers in genomic prediction, which turned the coefficient matrix into a dense matrix instead of a sparse matrix, making it a lot harder to evaluate the score functions. Originally, for the genomic prediction models, the variance of the genetic marker effects was derived from a previous estimate of the total genetic variance, where this genetic variance was commonly just divided by the number of genetic markers involved [14, 16, 71]. It was also proposed to use Gibbs sampling for variance component estimation, definitely when the data sizes were large [35, 70]. However, Gibbs sampling is not entirely automatic and the results depend on the choices made by the user. Although AI-REML also depends on the choice of the starting values of the variance components, algorithms exist which circumvent this problem by using a primary round of EM estimation before using AI-REML with as inital values the EM estimations obtained in the primary round [72]. Moreover the AI-REML algorithm is usually a lot faster than Gibbs sampling and it is not subject to Monte Carlo error as can be the case when using Monte Carlo estimators [73]. As such, this dissertation has as a goal to enable AI-REML estimation of variance components for large-scale data sets. Therefore, the next chapter will discuss some high performance computing techniques that can be applied for making it possible to perform an AI-REML estimation for the variance components of large-scale data, using the power of a supercomputing cluster.

5 A selection of high performance computing methods

5.1. Introduction

In the previous chapters, the computational aspects of the mixed model approach and variance component estimation are always emphasised to make clear that the research field of genomic prediction depends on advances in numerical computation methods as well as in computing power. It was already mentioned that numerical methods such as preconditioned conjugate gradient or Gauss-Seidel iterations for solving a system of equations have a huge impact on the applicability of genomic prediction. Moreover, for variance component estimation, the development of different iterative schemes for maximizing a likelihood function opened up the path for more complex models and the analysis of larger data sets. However, it should also be noted that many of the computational advances originated from the field of genomic prediction itself, showing that the computational problems encountered in genomic prediction are to a certain extent specific to the field, making it an interesting world not only for biologists, but also for computational scientists. This is also confirmed by the many research papers from the animal and plant breeding field that deal with the computational burdens encountered in genomic prediction [16, 34, 35, 36, 39, 45, 53, 55, 65, 69, 70].

It may thus not come as a surprise that from the moment the digital computer became available for researchers, it was used to help animal and plant breeders for evaluating their phenotypic records. The start of this computer-aided data-processing in a breeding application could be marked by the acquisition of an IBM 650 by Cornell University in 1956, when Henderson was research leader of the Dairy Records Processing Laboratory at this institute [74]. Henderson and his PhD students made intensive use of this machine because it enabled them to invert a 10×10 matrix in only 7 minutes compared to about three hours when it should have been done by hand. Using such computing power they were able to process data of 108,000 cows per year. This put the lab of Henderson at the forefront of animal

breeding, not only because of the availability of such a computer, but also due to their constant strive for improving the analytical methods, leading to increasing prediction accuracy of genetic merits of dairy bulls [7, 8, 45, 48]. As a result, Henderson was awarded the 1977 National Association of Animal Breeders Award from the American Dairy Association, where it was noted that Henderson's work helped increase milk production from 6,810 pounds per cow in 1950 to 13,612 in 1976 [74].

Computing power increased during the following years due to the invention of the integrated circuit in 1959 [75]. The evolution of computing power is adequately summarized by Moore's law, stating that the number of components per integrated circuit would double every year [32]. This law was conceived in 1965 as an observation of the evolution from 1959 to 1965, but had to be revised in 1975 to a doubling of the number of transistors per integrated circuit every two years [76]. Of course, such technological advances resulted in more powerful computers and the chip performance even doubled more rapidly because of a combined effect of the increased number of transistors and the fact that these transistors were of better quality and thus became faster. Numerous machines were built using integrated circuits and also the Dairy Records Processing Laboratory of Henderson updated its computing infrastructure frequently to enable processing data of more and more cows, leading to a total of about 650,000 cows under analysis in 1984.

Computers in those days were so big that they needed an entire room to fit in, but due to the introduction of the microprocessor in the beginning of the 1970s, more compact and portable personal computers were produced. Nowadays, microprocessors are so dense and powerful that our mobile phones can perform tasks faster than the gigantic computers of the 1960s. However, room-filling computer systems still exist for high-end purposes and are usually dubbed *supercomputers* to distinguish these systems from the personal computers such as laptops or desktops. The term supercomputer was already used in the 1960s for systems that achieved superior performance by applying innovative designs. One example is the CDC6600, which was 10 times faster than all other computers of that time because of switching from germanium to silicon transistors and using a refrigeration system to prevent the transistor from overheating [77]. Therefore, it is generally considered as the first successful supercomputer. From the late 1980s, starting with the conception of the Cray-2 machine in 1985, supercomputers are referred

to as machines consisting of several processors, while parallelism between these processing units can be exploited. The Cray-2 machine was the fastest machine in the world until 1990 and again it was used in animal breeding to estimate the variance components for an animal model with almost 75,000 effects [34].

From the 1990s, the number of processors that could be used in parallel on supercomputers increased dramatically and the current fastest supercomputer consists of more than 3 million CPU cores that can be run in parallel. This opened up the path to many different architectures and ways of exploiting this huge amount of processors for highly demanding computing tasks. Actually, even common laptops and desktops can be compared to the supercomputers of the 80s and the 90s as they also consist of several processing units that can run in parallel, using a shared memory block (the RAM memory in current computers). This is commonly referred to as multicore computing or symmetric multiprocessing and is still an active branch in the parallel computing field. However, the vast majority of the currently top-ranked supercomputers are cluster computers. The ranking of these supercomputers is based on a benchmark that measures the performance of a system when solving a dense system of equations and is publicly available in the so-called TOP500 list that is updated twice a year [78]. A cluster computer can be regarded as a large supercomputer, consisting of several loosely coupled computing units. These computing units are commonly referred to as computing nodes and they are coupled to each other through a local network. Different levels of coupling between these computing nodes result in different applicability of these computer clusters.

The most loosely coupled clustering is performed by means of the internet, thus restricting the amount of communication between the different computing nodes. Such a type of parallel computing is commonly known as grid computing and although many computing nodes can in this way be coupled to each other, it is not really suited for high performance computing due to the low bandwidth and high latency of the internet. However, it can be very useful for executing a large number of independent tasks, which do not require a lot of communication between the different computing nodes. This is commonly referred to as embarrassing parallelism and such applications are considered the easiest to be parrallelised. A striking example of grid computing is the “Great Internet Mersenne Prime Search" (GIMPS), where more than 100,000 participants share their computing power to search for

all possible Mersenne prime numbers, which are prime numbers that are one less than a power of two ($M_p = 2^p - 1$) [79]. This project resulted in the discovery of 14 Mersenne prime numbers of which also the largest known until now, $M_{57,885,161}$.

However, these grid computers usually are not regarded as supercomputers, because the computing units are not entirely available whenever necessary and the computing power can only be applied for performing many different tasks without much communication between the computing nodes. They are thus also not included in the TOP500 list, as they perform quite poorly on the benchmark used for compiling this list. Current supercomputers are mainly located at the same geographical location where the different computing nodes are connected to each other using a high-speed communication network. When the computing nodes each dispose of local memory that is not shared and they can only exchange information through message passing, the computer system is called a distributed system. In general, grid computers are a subset of distributed systems, but we will make a distinction between the two by the fact that distributed systems can be applied for so-called coarse-grained parallel tasks, which require some communication between the nodes, while grid computers almost do not communicate with each other and are only suited for embarrassingly parallel tasks. Systems optimised for distributed computing thus consist of several autonomous computing nodes disposing of local memory and connected to each other with a high-speed network to minimise the communication overhead.

The computing nodes of these distributed systems nowadays are high-end multicore processors, meaning that there are two types of parallelism that can be exploited in these systems. At first the coarse-grained parallelism between the different nodes can be utilised, where communication between the nodes is only possible using the interconnection network and should thus be limited to minimise the communication overhead. Secondly, the fine-grained parallelism inside each node can be applied, where each CPU core can access the shared memory of the computing node with high speed and can process in parallel a part of the task that is dedicated to the computing node. A widely used application programming interface (API) for shared memory multiprocessing programming is OpenMP, while for message passing between processes the standardized message passing interface (MPI) is freely available in the implementations MPICH and Open MPI, but also in commercial implementations as IntelMPI, Microsoft MPI and IBM

Platform MPI. Whenever MPI is used for message passing between nodes and OpenMP exploits the multicore parallellism on these nodes, it is common to denote this as a hybrid OpenMP/MPI implementation. These hybrid models are gaining popularity as they efficiently use the computing power of most currently available supercomputers.

Of course, parallel programming has its limitations, which should be taken into account when parallelising existing implementations for utilising the computing power of supercomputers efficiently. A well-known law in parallel computing is Amdahl's law, stating that the maximum speedup achievable by parallel programming is limited by the fraction of the program that is strictly serial and cannot be parallellised [80]. More formally, when α is the fraction of the program that is strictly serial, the time $T(n)$ the algorithm takes to finish when executed on P parallel threads is

$$T(P) = T(1)\left(\alpha + \frac{1}{P}(1-\alpha)\right). \tag{5.1}$$

As such the theoretical speedup achieved when being executed on n threads is

$$S(P) = \frac{T(1)}{T(P)} = \frac{1}{\alpha + \frac{1}{P}(1-\alpha)}. \tag{5.2}$$

Therefore, when for example 10% of an algorithm is strictly serial, the maximum attainable speedup is 10, no matter how many threads can be executed in parallel. Figure 5.1 shows the fact that the number of processors that can be efficiently applied in parallel depends greatly on the fraction of the program that can be parallelised. Moreover, increasing the number of parallel threads might result in a substantial overhead, due to the communication and synchronization between the threads, leading to a decrease in speedup whenever the number of applied threads surpasses a critical value.

Nonetheless, Amdahl's law supposes that the problem size is fixed, while an increase in computing power may lead to solving larger problems in the same amount of time. This was indicated by Gustafson's law assuming that the total amount of work to be done in parallel increases linearly with the number of processors [81]. As such, the achieved speedup by parallelism can be expressed as

$$S(P) = P - \alpha(P-1), \tag{5.3}$$

where P is the number of involved processors, linearly dependent on the

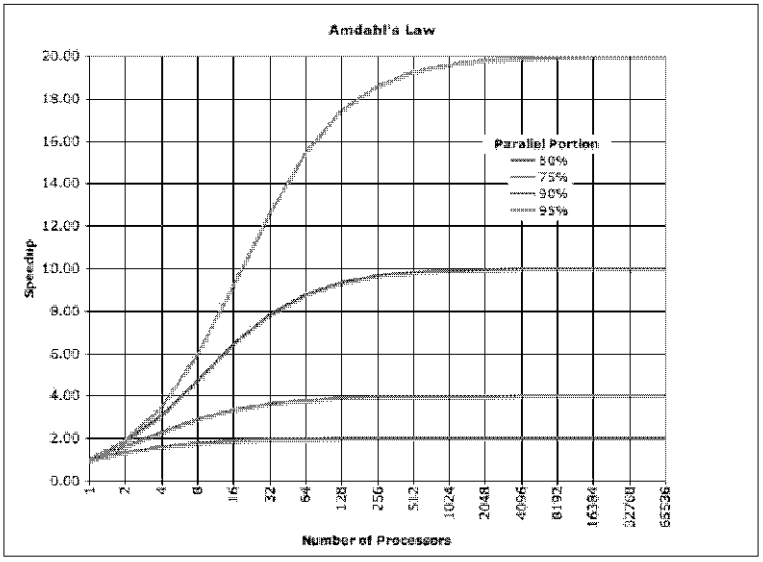

Figure 5.1: Graphical representation of Amdahl's law where speedup is depicted in function of the number of involved processors for programs with different fractions that can be parallelised.

problem size. As Amdahl's law states that parallel computing can only have a minor impact for fixed problem sizes, Gustafson's law actually increases the impact of parallel computing by stating that larger problems may always benefit from increased computing power, without letting the serial part of the program bring down the benefit of parallelisation. Parallelism is as such defined to be more fruitful for problems that can increase dramatically in size. Therefore, it can become extremely interesting for the genomic prediction field, where not only the number of data points is likely to increase due to the reduced cost of genotyping, but also the dimensionality of the data points may increase due to the larger number of genetic markers included in the analysis [42, 82].

For clarity, we would like to introduce the difference between high performance computing and big data processing. Big data recently gets a lot of attention as the digitization of data gathering leads to enormous amounts of data that can be analyzed to gain insights in processes or to predict future scenarios. The analysis of these data sets is usually tedious due to its size and not due to the processing algorithm. Therefore, the number of required computing cycles may not be the limiting part of the analysis, but overhead is mainly created by data movement, data processing and data

management. In contrast, high performance computing deals with problems that may not be as large as in big data processing, yet the actual computing cycles on the involved processors needed for processing the data are the most time-consuming aspect. The main difference between big data processing and high performance computing is the fact that for the latter the data is usually read in once and then stays stored in that same location, while for the former pieces of the data set are read in and analyzed frequently, sometimes by different processors, leading to many read and write operations. As discussed in the previous chapters, genomic prediction is based on solving large matrix equations and inverting matrices several times in an iterative cycle, while only changing some elements of the matrices. Therefore, the data can stay stored at a certain location and high performance computing methods can be applied for a more efficient processing of the data.

At Ghent university different centrally coordinated supercomputers were installed from 2008 to boost high performance computing research in several disciplines. One of them was the Gengar cluster, consisting of 194 computing nodes, containing 8 CPU cores per node and 16 GB of RAM, while being interconnected through a 4x DDR Infiniband network (20 GB/s). It was thus optimised for distributed high performance computing applications using MPI due to the fast interconnection network and it even reached the TOP500 list in 2008. In 2013 this system was replaced by an even more powerful supercomputer Delcatty, consisting of 160 compute nodes with 16 cores per node and 64 GB of RAM per node, interconnected with an FDR Infiniband network (56 GB/s). These two clusters were mainly used in this dissertation as they were configured especially for the needs of this research. As already said, genomic prediction can benefit from distributed computing as it mainly involves solving large systems of equations and inverting large matrices. Therefore, the aggregated memory of the different computing nodes is used to store the matrices, while the computing power of all the CPU cores can be applied for performing operations on these matrices. The benchmark for the TOP500 list of supercomputing sites is also based on solving large systems of linear equations and thus the distributed computing approach will also be portable to many high performance computing infrastructures.

Next to the distributed computing methods, the genomic prediction framework as elaborated on in the previous chapters can also benefit from sparse matrix algebra, where matrices filled with many zeroes are stored and processed more efficiently. Technically speaking, sparse linear algebra is not

specifically a high performance computing method as it can also be used on a single processor. However, as we intend to use it in combination with distributed computing techniques, some important properties of sparse matrix algebra are discussed in this chapter. The remainder of this chapter is thus organised as follows:

- Section 5.2 will explain the main properties of distributed computing with an emphasis on linear algebra and matrix operations. Parallelisation and optimisation of these operations will be discussed using some basic examples.
- Section 5.3 will present some properties of sparse matrix algebra, together with efficient methods for dealing with sparse matrices. Emphasis is put on solving a sparse system of equations and inverting a sparse matrix.
- Section 5.4 will review the usefulness of both distributed computing methods and sparse matrix algebra in a genomic prediction framework and will briefly discuss some computational advances throughout the history of genomic prediction.

5.2. Distributed computing

5.2.1. General overview

Distributed computing refers to the efficient use of distributed systems for tackling large-scale problems, which become computationally impractical to be processed on a single computing node. These distributed systems consist of computers that are interconnected through a, preferably high-speed, network, allowing for communication between the computing nodes through message passing. The messages can be certain objects that need to be transferred to another node, ranging from simple integer values to whole arrays of complex datatypes. But the messages can also be signals allowing the processes to coordinate the tasks that have to be performed by each processor. An important aspect of distributed computing is that each computing node only has access to its own local memory and no memory is shared among the different computing nodes. It is thus vital that each computing node reads in the correct information from an input file and that

the messages passed between the nodes contain the correct information for the receiving node.

To facilitate the message passing between the different nodes, the standardized and portable Message Passing Interface was released in 1994 [83]. The conception of this standardized interface was a joint collaboration between members of more than 40 organizations, including major vendors of concurrent computers, universities, government laboratories and industry. The goal of such an interface was to design an application programming interface, which allows for efficient communication, presents convenient C and Fortran 77 bindings for the interface, assumes a reliable communication interface, where the user need not cope with communication failures and defines an interface that can be implemented on many vendor's platforms. The successful design of this standard interface led to multiple implementations of the standard in open-source distributions (MPICH, Open MPI) as well as commercial implementations from Intel, HP or IBM. Nowadays, there is already a third version of the MPI standard available (MPI-3), including several extensions to the original version, keeping MPI still very relevant in any supercomputing application.

The release of such a standard directly led to the development of a library that makes use of MPI for performing several linear algebra operations on distributed systems. This library was called ScaLAPACK, an abbreviation for Scalable LAPACK, where LAPACK is an acronym for linear algebra package. LAPACK was conceived in 1992 as an attempt to make linear algebra operations run more efficiently on shared-memory parallel processors by reorganizing the algorithms to minimize data transportation through the memory hierarchy of a processor. In fact, all machines (not just supercomputers) have a hierarchy of memory levels, for example, with registers at the top, followed by cache, RAM memory, and finally disk storage at the bottom. At the top of the hierarchy, the memory is smaller, faster and more expensive, making it ideal to perform a lot of small-scale operations, while at the bottom we prefer to have a small number of large-scale operations. The key to the efficient use of this memory hierarchy is dividing a matrix into smaller blocks which fit in the top level memory and performing the algebraic operations on the entire matrix in a blockwise manner. In this way vendors of processors can optimise the standard library for matrix operations by choosing an appropriate size of the blocks so they can fit in the different memory levels, while minimising data transportation in the

memory hierarchy.

LAPACK provides routines for solving linear equations and least squares problems, factorizing matrices, inverting matrices and calculating singular value and eigenvalue decompositions. These routines are all based as much as possible on matrix-matrix operations as defined in level 3 BLAS. BLAS are the Basic Linear Algebra Subprograms that provide standard building blocks for vector-vector (level 1), matrix-vector (level 2) and matrix-matrix (level 3) operations. The level 1 BLAS were already conceived in 1979 [84], but level 2 and 3 BLAS only saw the daylight ten years later, because these made it possible to exploit the memory hierarchy of the processing units that were invented around that time. BLAS and LAPACK are still part of most mathematical software such as MATLAB, Mathematica and R. There also exist several vendor-optimized implementations available in Intel MKL for Intel processors and ACML for AMD processors, but there are also free alternatives such as GotoBLAS, OpenBLAS and ATLAS, which provides optimized BLAS versions for different types of processors.

As BLAS and LAPACK form the basis of ScaLAPACK, we will first describe how they achieve high performance. Afterwards, it will be explained how matrices can be processed in a distributed-memory setting and finally we will show how parallelism can be exploited in these distributed memory systems, leading the the distributed-memory parallel BLAS (PBLAS) and ScaLAPACK.

5.2.2. Blockwise algebraic operations

It has been presented above that performing algebraic operations in a blockwise manner can exploit the memory hierarchy in current processing units. As this is a vital step in understanding distributed computing we will present two examples where blockwise computations can lead to a significant gain in performance. One example , the matrix-matrix multiplication, is a level 3 BLAS routine, while the other example, an LU decomposition, is a LAPACK routine based on level 3 BLAS routines.

At first, we will handle a simple matrix-matrix multiplication for 2 matrices $\mathbf{A}$ and $\mathbf{B}$, with respective dimensions $m \times l$ and $l \times n$. The elements c_{ij} of

the product $\mathbf{C} = \mathbf{AB}$ with dimensions $m \times n$ are given by

$$c_{ij} = \sum_{k=1}^{l} a_{ik} b_{kj} \,.$$

Let us now split up all dimensions in two parts, where $m = m_1 + m_2$, $l = l_1 + l_2$ and $n = n_1 + n_2$, so each of the involved matrices is split up in 4 blocks:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{(1,1)} & \mathbf{A}_{(1,2)} \\ \mathbf{A}_{(2,1)} & \mathbf{A}_{(2,2)} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} \mathbf{B}_{(1,1)} & \mathbf{B}_{(1,2)} \\ \mathbf{B}_{(2,1)} & \mathbf{B}_{(2,2)} \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} \mathbf{C}_{(1,1)} & \mathbf{C}_{(1,2)} \\ \mathbf{C}_{(2,1)} & \mathbf{C}_{(2,2)} \end{bmatrix} ,$$

where the dimension of $\mathbf{A}_{(1,1)}$ is $m_1 \times l_1$, of $\mathbf{A}_{(1,2)}$ is $m_1 \times l_2$, of $\mathbf{A}_{(2,1)}$ is $m_2 \times l_1$, of $\mathbf{A}_{(2,2)}$ is $m_2 \times l_2$ and similarly for the other matrices. The elements of $\mathbf{C}_{(1,1)}$ can be expressed as, for $1 \leq i \leq m_1$, $1 \leq j \leq n_1$:

$$\begin{aligned} \left[\mathbf{C}_{(1,1)}\right]_{ij} = c_{ij} &= \sum_{k=1}^{l} a_{ik} b_{kj} = \sum_{k=1}^{l_1} a_{ik} b_{kj} + \sum_{k=l_1+1}^{l_1+l_2} a_{ik} b_{kj} \\ &= \left[\mathbf{A}_{(1,1)} \mathbf{B}_{(1,1)}\right]_{ij} + \left[\mathbf{A}_{(1,2)} \mathbf{B}_{(2,1)}\right]_{ij} \\ &= \left[\mathbf{A}_{(1,1)} \mathbf{B}_{(1,1)} + \mathbf{A}_{(1,2)} \mathbf{B}_{(2,1)}\right]_{ij} . \end{aligned}$$

For the other blocks, we can derive similar expressions showing that for matrix multiplications the blocks can be treated as if they were elements of a matrix:

$$\begin{aligned} \begin{bmatrix} \mathbf{C}_{(1,1)} & \mathbf{C}_{(1,2)} \\ \mathbf{C}_{(2,1)} & \mathbf{C}_{(2,2)} \end{bmatrix} &= \begin{bmatrix} \mathbf{A}_{(1,1)} & \mathbf{A}_{(1,2)} \\ \mathbf{A}_{(2,1)} & \mathbf{A}_{(2,2)} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{(1,1)} & \mathbf{B}_{(1,2)} \\ \mathbf{B}_{(2,1)} & \mathbf{B}_{(2,2)} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_{(1,1)} \mathbf{B}_{(1,1)} + \mathbf{A}_{(1,2)} \mathbf{B}_{(2,1)} & \mathbf{A}_{(1,1)} \mathbf{B}_{(1,2)} + \mathbf{A}_{(1,2)} \mathbf{B}_{(2,2)} \\ \mathbf{A}_{(2,1)} \mathbf{B}_{(1,1)} + \mathbf{A}_{(2,2)} \mathbf{B}_{(2,1)} & \mathbf{A}_{(2,1)} \mathbf{B}_{(1,2)} + \mathbf{A}_{(2,2)} \mathbf{B}_{(2,2)} \end{bmatrix} . \end{aligned}$$

This can also be proven for any number of blocks, as long as the matrix dimensions of the blocks of the matrices agree [85]. So when the dimension m is split up in q parts, n is split up in r parts and l is split up in s parts, the different blocks of $\mathbf{C}$ can be computed as, for $1 \leq \alpha \leq q$, $1 \leq \beta \leq r$

$$\mathbf{C}_{(\alpha,\beta)} = \sum_{\gamma=1}^{s} \mathbf{A}_{(\alpha,\gamma)} \mathbf{B}_{(\gamma,\beta)} \,.$$

In this way a blocksize can be chosen so that the different blocks all fit in the top level of the memory hierarchy and thus operations on these blocks can be performed very efficiently. As an example, we give some experimental timings for a matrix-matrix multiplication of two square matrices of dimensions $n \times n$ with varying blocksizes $b \times b$ in Table 5.1 and compare this with a naive implementation of matrix-matrix multiplication without splitting the matrices into blocks. Note that when a matrix is split up in blocks, all blocks are square and have an equal size across the involved matrices. It can be seen that the blockwise algorithm clearly outperforms the naive implementation with a maximum speedup of almost 5 for the largest matrices. Moreover, the size of the blocks also plays a role as too small blocks lead to a suboptimal use of the top level memory and too large blocks will have to use partly the second level memory which is slightly slower. Next to this, we compare these implementations with the Intel version for matrix-matrix multiplication as implemented in the Intel Math Kernel Library (MKL) level 3 BLAS. This library provides optimised versions of BLAS and LAPACK for Intel processors and their specific memory hierarchy. A remarkable 16-fold decrease in runtime is achieved by using the Intel-optimised version for the largest matrices, which is probably due to the fact that some sort of recursive blocking is used to optimise the use of all memory levels of the processing unit. Unfortunately, Intel MKL is commercial software and only free under an academic license.

Table 5.1: Runtimes (in seconds) for matrix-matrix multiplications of two equally sized square matrices of dimension $n \times n$ on a single Intel Nehalem processor for different algorithms (averaged over 10 runs)

Matrix size	Naive	Blockwise algorithm with blocksize			Intel MKL
(n)	algorithm	4×4	16×16	64×64	version
2,048	89.9	30.3	24.3	29.2	1.9
4,096	1,080.9	271.2	229.6	231.5	14.5
8,192	9,002.1	2,387.1	1,831.8	1,844.6	114.9

The second example that we will discuss is the LU decomposition of a matrix by Gaussian elimination, because such a decomposition is commonly used for solving linear equations or inverting a matrix. The LU decomposition factorises the matrix in a lower and upper triangular matrix and solving a linear equation is then transformed into solving 2 linear equations, which can be performed using forward and backward substitution with a complexity

$O(n^2)$ (with n the number of linear equations), compared to a complexity $O(n^3)$ of the LU decomposition:

$$\mathbf{Ax} = \mathbf{y} \Leftrightarrow \mathbf{LUx} = \mathbf{y}\,.$$

1. Solve $\mathbf{Lw} = \mathbf{y}$ with forward substitution.
2. Solve $\mathbf{Ux} = \mathbf{w}$ with backward substitution.

The algorithm for Gaussian elimination is simple in nature and can be performed column-by-column with elementary row operations. For a square matrix $\mathbf{A}$ of order n the algorithm is described in Algorithm 1, where memory requirements are reduced by overwriting the lower and upper triangular part of $\mathbf{A}$ by resp. $\mathbf{L}$ and $\mathbf{U}$

Algorithm 1 Calculate LU decomposition of a square matrix $\mathbf{A}$ of order n

```
1: procedure LUdecomp
2:     for i ← 1 to n step 1 do
3:         for j ← i + 1 to n step 1 do
4:             A(j, i) ← A(j, i)/A(i, i)
5:         end for
6:         for j ← i + 1 to n step 1 do
7:             for k ← i + 1 to n step 1 do
8:                 A(j, k) ← A(j, k) − A(j, i) ∗ A(i, k)
9:             end for
10:        end for
11:    end for
12: end procedure
```

For large matrices this algorithm becomes inefficient, as it only applies vector operations, where matrix operations as implemented in the level 3 BLAS are more efficient than the vector operations in the level 2 BLAS [86]. Therefore, the algorithm can also be performed in a blockwise manner, where we can make use of the following expressions:

$$\begin{aligned}\mathbf{M} = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix} &= \begin{bmatrix}\mathbf{L}_A & \mathbf{0}\\ \mathbf{L}_C & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{S}\end{bmatrix}\begin{bmatrix}\mathbf{U}_A & \mathbf{U}_B\\ \mathbf{0} & \mathbf{I}\end{bmatrix} \\ &= \begin{bmatrix}\mathbf{L}_A\mathbf{U}_A & \mathbf{L}_A\mathbf{U}_B\\ \mathbf{L}_C\mathbf{U}_A & \mathbf{L}_C\mathbf{U}_B + \mathbf{S}\end{bmatrix},\end{aligned} \tag{5.4}$$

where $\mathbf{A} = \mathbf{L}_A\mathbf{U}_A$ is the LU decomposition of the $\mathbf{A}$ block matrix and $\mathbf{S}$ is also known as the Schur complement. $\mathbf{L}_C$ and $\mathbf{U}_B$ are mostly not triangular matrices, but are a submatrix of a larger triangular matrix. The complete LU decomposition of the above-defined matrix can be obtained by calculating the LU decomposition of $\mathbf{S} = \mathbf{L}_S\mathbf{U}_S$ and inserting it in Eq. 5.4:

$$\begin{aligned}\mathbf{M} = \begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix} &= \begin{bmatrix}\mathbf{L}_A & \mathbf{0}\\ \mathbf{L}_C & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{L}_S\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{U}_S\end{bmatrix}\begin{bmatrix}\mathbf{U}_A & \mathbf{U}_B\\ \mathbf{0} & \mathbf{I}\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{L}_A & \mathbf{0}\\ \mathbf{L}_C & \mathbf{L}_S\end{bmatrix}\begin{bmatrix}\mathbf{U}_A & \mathbf{U}_B\\ \mathbf{0} & \mathbf{U}_S\end{bmatrix}.\end{aligned}$$

The LU decomposition of $\mathbf{S}$ can of course also be performed in a blockwise manner and thus an improved iterative blockwise algorithm, making use of matrix operations instead of vector operations is described in Algorithm 2.

Algorithm 2 Calculate LU decomposition of a square matrix $\mathbf{M}$ of order n in blocks of size b

1: **procedure** BlockLUdecomp
2: **for** $i \leftarrow 1$ **to** n **step** $b+1$ **do**
3: $ib = i + b$
4: ▷ We use the non-blocked algorithm for calculating the LU decomposition of the block with dimension $b \times b$.
5: LUdecomp $(\mathbf{M}(i:ib, i:ib))$
6: ▷ $\mathbf{L}_i$ and $\mathbf{U}_i$ denote the lower and upper triangular part of $\mathbf{M}(i:ib, i:ib)$ after LU decomposition.
7: $\mathbf{M}(i:ib, ib+1:n) \leftarrow \mathbf{L}_i^{-1}\mathbf{M}(i:ib, ib+1:n)$
8: ▷ by forward substitution
9: $\mathbf{M}(ib+1:n, i:ib) \leftarrow \left(\mathbf{U}_i^{-T}\mathbf{M}(ib+1:n, i:ib)^T\right)^T$
10: ▷ by forward substitution
11: $\mathbf{M}(ib+1:n, ib+1:n) \leftarrow \mathbf{M}(ib+1:n, ib+1:n) - \mathbf{M}(ib+1:n, i:ib) * \mathbf{M}(i:ib, ib+1:n)$
12: **end for**
13: **end procedure**

Again a blocksize can be chosen so the blocks that are decomposed fit in the top level memory and decomposition of these blocks can be performed efficiently, while the forward substitutions and matrix-matrix multiplications are performed with the aid of the optimised level 3 BLAS, which inherently

uses blockwise operations. In Table 5.2 we compare the naive implementation as in Algorithm 1 with the blockwise LU decomposition using optimised level 3 BLAS and ultimately it is again compared with the MKL version of an LU decomposition. As for matrix-matrix multiplication we again see that blocksizes should not be too small or too big, but in this case the optimal blocksize is somewhat larger as for matrix-matrix multiplication because for the latter, 3 blocks should be stored simultaneously in the top level memory as opposed to only one block for LU decomposition. In this case the optimised Intel MKL version does not outperform significantly our blockwise implementation as both versions are based on the same optimised level 3 BLAS routines of the Intel MKL.

Table 5.2: Runtimes (in seconds) for an LU decomposition of a square matrix of dimension $n \times n$ on a single Intel Nehalem processor for different algorithms (averaged over 10 runs)

Matrix size	Naive	Blockwise algorithm with blocksize			Intel MKL
(n)	algorithm	8×8	64×64	512×512	version
4,096	29.0	8.0	6.0	7.8	5.3
8,192	237.1	62.7	44.4	51.5	40.0
12,288	1,123.52	234.5	150.2	174.1	132.5

For simplicity both algorithms described above do not take into account the partial pivoting that was also implemented in all algorithms. This partial pivoting is introduced to counteract the numerical instability that can occur when a diagonal element becomes very small. As a remedy, we always look for the element with the largest absolute value in a column i, and swap the row where the element belongs to with row i. In this way, actually a permuted LU decomposition is calculated: $\mathbf{A} = \mathbf{PLU}$, with $\mathbf{P}$ a permutation matrix, but this has only minor implications on the complexity of the algorithm.

5.2.3. Distribution of matrices over the local memories of the different computing nodes

The algorithms introduced in the previous section are very general and the blocks of the matrices that are processed in each iteration might even be read in from the lowest memory level (disk storage) and the solutions written back

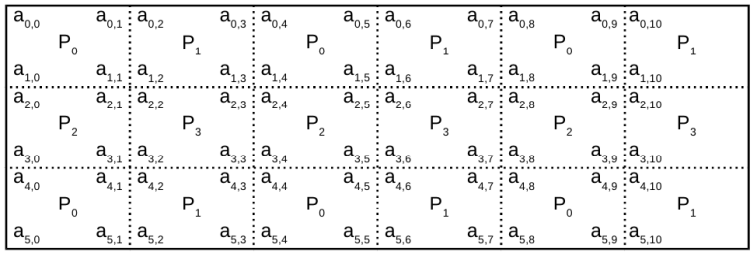

Figure 5.2: Schematic representation of a 2 dimensional block cyclic data layout of a 6×11 matrix with elements $a_{i,j}$ for a 2×2 process grid of 4 processes P_s and blocks of size 2×2

to this disk storage. However, reading from and writing to this disk storage is very slow and so it is desired that this kind of memory is not used for basic operations such as matrix multiplications, because reading and writing the data then becomes the bottleneck of the algorithm instead of the actual computations. Therefore, it is vital to be able to store the entire matrix at least in RAM memory which can be read and written to very rapidly. RAM memory is mostly limited in size and it should thus always be kept in mind what size of matrix can be stored in the RAM memory of the computing node. This of course depends on the type of elements in the matrix, but usually we would like to store the elements as 64-bit floating point numbers (usually referred to as `doubles`) for optimal numerical accuracy. For instance when 16 GB of RAM is available on a machine, the maximum size of a square matrix of `doubles` that can be stored and processed is 44,721. For processing larger matrices in an efficient way, one must then look for a machine with more RAM memory installed, or distribute the matrix over the local memories of different interconnected nodes. The latter is considered in this dissertation as such interconnected machines are already widely available and it has the benefit that the computing power of each of these computing nodes may be used in parallel to reduce computation time as well.

There are several ways of distributing a matrix over the local memories of different computing nodes, but the recommended strategy is the so-called two-dimensional block cyclic layout. This data layout has shown to perform well in terms of scalability, load balance and communication properties [87, 88]. Such a distribution starts with splitting the original $m \times n$ matrix into rectangular submatrices of size $m_b \times n_b$ and mapping the available p processes in a rectangular $p_r \times p_c$ grid. An example of the

two-dimensional block cyclic data layout can be found in Figure 5.2, for a matrix of size 6×11, with blocksize 2×2, distributed on a 2×2 process grid. Such a data distribution has several advantages. First of all the blockwise distribution enables the use of level 2 and level 3 BLAS to perform algebraic operations on the submatrices present in the memory of each process. Secondly, when looking at the LU decomposition operations are performed on the matrix from left to right and from the top down, and thus the cyclic distribution offers a good load balance as opposed to a non-cyclic distribution where for example P_0 would no longer contribute after n/P steps. Finally, the two-dimensional data distribution offers possibilities to exploit parallelism both column-wise and row-wise.

5.2.4. Exploiting parallelism in linear algebraic operations

Distributed computing does not only have as a benefit that larger matrices can be processed, but it can also make use of the aggregated computing power of the involved computing nodes. Therefore, some parallelism should be exploited in the different algebraic operations. If we look at matrix-matrix multiplications, we have seen that each block of the matrix product can be formed, for $1 \leq \alpha \leq q$, $1 \leq \beta \leq r$, as

$$\mathbf{C}_{(\alpha,\beta)} = \sum_{\gamma=1}^{s} \mathbf{A}_{(\alpha,\gamma)} \mathbf{B}_{(\gamma,\beta)} \, .$$

The SUMMA method (Scalable Universal Matrix Multiplication Algorithm) is a widely-used algorithm for exploiting the parallelism of a matrix-matrix multiplication, based on this blocking of the matrices [89]. The algorithm is again quite simple in nature, but has shown to be highly efficient when memory use per node is kept constant. The pseudo-code for this algorithm is shown in Algorithm 3, where this algorithm can be executed by each process in the process grid in parallel. The broadcasts mentioned in the algorithm denote that the process possessing the matrix block sends the block, while the other processes in the process row or process column receive the matrix block.

The algorithm can be summarised as follows. The processes calculate in parallel the blocks of $\mathbf{C}$ stored in their local memory by receiving the blocks

Algorithm 3 Calculate matrix multiplication $\mathbf{C} = \mathbf{AB}$ with $\mathbf{A}$ and $\mathbf{B}$ of resp. dimensions $m \times l$ and $l \times n$ distributed over a $p_r \times p_c$ grid with blocks of dimension $b \times b$

```
1: procedure SUMMA
2:     for i ← 1 to m/b step 1 do
3:         if this process contains a block C_(i,*) then
4:             for j ← 1 to n/b step 1 do
5:                 if this process contains the block C_(i,j) then
6:                     for k ← 1 to l/b step 1 do
7:                         broadcast A_(i,k) over process row
8:                         broadcast B_(k,j) over process column
9:                         C_(i,j) = C_(i,j) + A_(i,k)B_(k,j)
10:                    end for
11:                end if
12:            end for
13:        end if
14:    end for
15: end procedure
```

of $\mathbf{A}$ and $\mathbf{B}$ necessary for computing the block of $\mathbf{C}$ in an iterative way. This parallelism based on a two dimensional block cyclic distribution of the matrix is also exploited in the parallel BLAS (PBLAS) routines. These routines form the basis of ScaLAPACK, which is a high-performance implementation of the LAPACK library for distributed systems. The PBLAS and ScaLAPACK routines make use of the Basic Linear Algebra Communication Subprograms (BLACS) that arrange the message passing between the nodes. Such a standard communication library enables portability between systems using different libraries for message passing. In this way, ScaLAPACK and PBLAS offer a standard solution for efficiently processing large-scale matrices on distributed systems.

5.3. Sparse matrices

The previous section elaborated on distributed computing methods for linear algebraic operations on large matrices. Sometimes the use of distributed systems can be avoided when a matrix contains a lot of zeroes and is only sparsely filled with non-zero values. Such a matrix is called sparse opposed to a dense matrix, where a matrix is generally treated as sparse whenever

less than 10% of the values are non-zero. The advantage of the sparsity of a matrix is the fact that zero values do not contribute in a lot of algebraic operations, as zero is the identity element in additions and subtractions and it is the absorbing element in multiplications. Thus, for instance, in a matrix-matrix multiplication, the zero elements of the factors can be ignored. Consequently, only the non-zero values of the factors should be stored and the algebraic operation can be performed by iterating only over the non-zero values.

There are BLAS available for sparse matrices, although they only contain routines for sparse vector operations, sparse matrix/dense vector operations and sparse matrix/dense matrix operations. This is partly because the previously explained method of blockwise operations is not applicable for most sparse matrices as for an arbitrarily chosen blocksize the number of non-zero elements in these blocks may vary dramatically, leading to an inefficient use of the memory hierarchy. For the factorisation of a matrix, solving linear equations and inverting a matrix, however, algorithms exist that are optimised for sparse matrices, which we will touch upon in this section. These algorithms are implemented in some widely used software packages such as PARDISO [90], MUMPS [91], HSL [92] and SuperLU [93]. They all offer various routines for sparse matrices based on slightly different implementations. In this dissertation, it was chosen to use PARDISO, as it offered the solutions suited for the problems encountered in this research and also because the developing team was very open for collaboration.

Some of the basic concepts of sparse matrix algebra will be introduced in the following sections, but for more details, we would like to refer to a book about solving sparse linear systems by Saad [94] and a very informative review paper by Liu [95].

5.3.1. Storage format

Where a dense matrix is most efficiently stored as an array containing the consecutive elements of the matrix when running through it column- or row-wisely from the top down and from left to right, there are several ways for storing a sparse matrix. The simplest way is to store the non-zero values together with their row and column coordinates, which enables the construction of the matrix in an easy way. However, a more compressed form that still allows for efficient arithmetic and algebraic operations is the

Compressed Row/Column Storage (CRS or CCS) format. The CRS format is also used by PARDISO and thus will also be used further on in this dissertation, but the CCS format is equivalent to the CRS format.

The coordinate format requires 3 arrays with the number of elements equal to the number of non-zeroes in the sparse format. These arrays are the array with the non-zero values (`val`), the array with the column indices of these non-zero values (`col`) and the array with the row indices of these non-zero values (`row`). Compared to the coordinate format, the CRS format compresses the array of the row indices `row` to an array with as size the number of rows in the matrix incremented by one (`row_ptr`). This is achieved by sorting the non-zero values row-wisely in the `val` array with their column indices at the same position in the `col` array. The `row_ptr` array contains at every position i the index where row i starts in the `val` and `col` arrays. The first element of the `row_ptr` array is always one and the last element (position $m + 1$, with m the number of rows of the matrix) is always the number of non-zero values incremented by one. As an example, if we look at the following sparse matrix

$$\begin{bmatrix} 10 & 0 & 5 & 0 \\ 2 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 4 & 0 & 1 & 11 \end{bmatrix},$$

the different arrays of the CRS format have the following elements:

$$\begin{aligned} \texttt{val} &= [10, 5, 2, 7, 4, 1, 11] \\ \texttt{col} &= [1, 3, 1, 2, 1, 3, 4] \\ \texttt{row\_ptr} &= [1, 3, 5, 5, 8] \ . \end{aligned}$$

Note that in this example we followed a one-based indexing, so when zero-based indexing is used the values of `col` and `row_ptr` should be reduced by one.

5.3.2. Solving a sparse system of linear equations

For solving a sparse system of linear equations, just as for solving a dense system of linear equations, two options exist, namely using direct solving

methods based on a matrix decomposition or using iterative solving methods applying successive approximations of an initial guess to converge towards the exact solution. These iterative methods, like the conjugate gradient method, have been around since the 1950s and are also applied for solving dense systems of linear equations. Because most iterative methods only need matrix-vector operations, the sparsity of the system can efficiently be exploited. However, a downside of the iterative methods is the fact that for each solution of a system with a similar coefficient matrix, the iterative cycle has to be repeated. Moreover, the behavior of iterative solvers can be unpredictable and they are mostly not robust for small perturbations [94]. Therefore, in the 1970s, a lot of research was oriented towards efficient sparse direct solvers, mainly initiated by electrical engineers for the design of electrical networks.

A first step for direct solving routines is computing the LU decomposition of the coefficient matrix of the system of equations. An obstacle that had to be overcome, was the fact that the LU decomposition of a sparse matrix could become densely filled with non-zero values. Such fill-in can be reduced by reordering or permuting the sparse matrix. A very simple but informative example is the LU decomposition of the following sparse matrix

$$\begin{bmatrix} 10 & 3 & 2 & 8 & 9 \\ 2 & 15 & 0 & 0 & 0 \\ 5 & 0 & 12 & 0 & 0 \\ 6 & 0 & 0 & 16 & 0 \\ 8 & 0 & 0 & 0 & 15 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0.2 & 1 & 0 & 0 & 0 \\ 0.5 & -0.1 & 1 & 0 & 0 \\ 0.6 & -0.1 & -0.1 & 1 & 0 \\ 0.8 & -0.2 & -0.2 & -0.7 & 1 \end{bmatrix} \begin{bmatrix} 10 & 3 & 2 & 8 & 9 \\ 0 & 14.4 & -0.4 & -1.6 & -1.8 \\ 0 & 0 & 11 & -4.2 & -4.7 \\ 0 & 0 & 0 & 10.5 & -6.2 \\ 0 & 0 & 0 & 0 & 2.5 \end{bmatrix}, \quad (5.5)$$

where the factors are two completely filled triangular matrices. This fill-in can easily be avoided by switching the first and last row and column of the matrix:

$$\begin{bmatrix} 15 & 0 & 0 & 0 & 8 \\ 0 & 15 & 0 & 0 & 2 \\ 0 & 0 & 12 & 0 & 5 \\ 0 & 0 & 0 & 16 & 6 \\ 9 & 3 & 2 & 8 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0.6 & 0.2 & 0.2 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} 15 & 0 & 0 & 0 & 8 \\ 0 & 15 & 0 & 0 & 2 \\ 0 & 0 & 12 & 0 & 5 \\ 0 & 0 & 0 & 16 & 6 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad (5.6)$$

where the factors now have the same sparsity pattern as the original matrix.

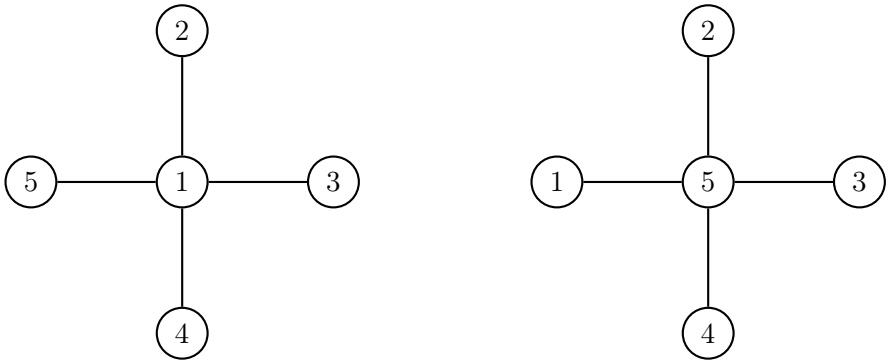

Figure 5.3: The graph representation of the matrices in Eq. (5.5) (left) and (5.6) (right).

Finding the best ordering for minimising fill-in in a more general case is an NP-complete problem and thus can only be approximated by heuristic methods. Algorithms that try to find the best fill-reducing ordering are all based on a graph representation of the sparse matrix. For a matrix with a symmetric structure this is an undirected graph, while otherwise a directed graph is used. In this section we will assume that the matrices will be symmetrical in structure and for these matrices, the nodes of the graph representation are the rows/columns of the matrix and an edge between node i and node j is drawn when element (i, j) of the matrix is not zero. The graph representations of both matrices are shown in Figure 5.3. The most simple, but well-performing algorithm for minimizing the fill-in of a sparse matrix when being factorised, is the so-called Cuthill-McKee algorithm, which starts at a node with the lowest degree and then orders all its neighbours from lowest to highest degree [96]. The algorithm iteratively goes through all the nodes of the graph using a breadth-first traversal, visiting first all the neighbours at a certain level before moving on to a next level of neighbours.

Actually, before the Cuthill-McKee algorithm was conceived, a more greedy approach was used, called the minimum degree algorithm [97]. This algorithm performs at each step of the Gaussian elimination procedure some row and column permutations to minimise the number of off-diagonal non-zeroes in the pivot row and column. Several enhancements and variations of this algorithm have been proposed, whereof the approximate minimum degree algorithm is probably the most popular because of its high speed and accuracy [98]. Another algorithm for finding the optimal fill-reducing ordering is the nested dissection algorithm based on recursive graph partitioning [99].

This algorithm is particularly used for symmetric positive definite matrices and even distributed-memory implementations of this algorithm, such as ParMETIS [100] and PT-Scotch [101], exist for handling very large sparse matrices.

Next to the continuous reduction in fill-in due to the improvement of the fill-reducing ordering algorithms, the direct solution method for sparse linear systems gained a significant boost by the development of a frontal solver in 1970 [102]. Such a frontal solver combines several non-zero elements in so-called frontal matrices that are efficiently partially factorized by dense matrix operations. An improvement to this method was the multifrontal method, making use of the fact that several frontal matrices could be processed independently of each other, based on a tree structure for detecting dependencies between frontal matrices [103]. This cleared the path for parallel processing of the frontal matrices and, furthermore, instead of factorising the sparse matrix column-by-column, several columns could be treated together in the supernodal form of the multifrontal method. This supernodal multifrontal method is the basis of most modern implementations of a direct solver for a sparse systems of linear equations and it is very efficient because it can use level 3 BLAS to perform algebraic operations on the supernodes. As a detailed description of this method is outside the scope of this dissertation, we refer the interested reader to the review paper by Liu [95].

5.3.3. Inverting a sparse matrix

The inverse of a sparse matrix is in most cases no longer a sparse matrix. For instance, if we again look at the sparse matrix from Eq. (5.6), then the inverse of this matrix can be computed as given by

$$\begin{bmatrix} 0.398 & 0.083 & 0.259 & 0.233 & -0.621 \\ 0.11 & 0.094 & 0.086 & 0.076 & -0.207 \\ 0.092 & 0.023 & 0.155 & 0.065 & -0.172 \\ 0.276 & 0.069 & 0.216 & 0.256 & -0.517 \\ -0.552 & -0.138 & -0.431 & -0.388 & 1.034 \end{bmatrix}, \tag{5.7}$$

which is a completely dense matrix. For completeness, we would like to remind the reader that permutations do not offer less fill-in in the inverse

matrix as is the case for the LU factors, because the inverse of a permutation matrix is equal to its transpose and so

$$\left(\mathbf{PAP}'\right)^{-1} = \mathbf{PA}^{-1}\mathbf{P}',$$

with $\mathbf{P}$ a permutation matrix.

However, sometimes it might be useful to calculate a subset of the elements of the inverse of a sparse matrix. For example, when having to calculate the trace of the product of a symmetric matrix with the inverse of its derivative towards a certain parameter, we only have to calculate the elements of this inverse that are non-zero in the original matrix. Such a subset can be calculated in a recursive manner by the Takahashi equations, which were already derived in 1973 [104]. The beauty of these equations lies in their simplicity and efficiency and, therefore, they are presented here for calculating a subset of the elements of the inverse of a sparse symmetric matrix. Assume $\mathbf{A}$ is a symmetric sparse invertible matrix of dimension $n \times n$ and let us denote its inverse by $\mathbf{Z}$. Because $\mathbf{A}$ is a symmetric matrix, it can be factorised as $\mathbf{LL}'$ by the Cholesky decomposition, with $\mathbf{L}$ a lower triangular matrix. As such,

$$\mathbf{ZA} = \mathbf{I}_n \Leftrightarrow \mathbf{ZLL}' = \mathbf{I}_n \Leftrightarrow \mathbf{ZL} = \mathbf{L}'^{-1}$$

and when defining $\mathbf{D} = \text{diag}\left(\mathbf{L}\right)$, a diagonal matrix with as diagonal elements the same elements as on the diagonal of $\mathbf{L}$, we can add $\mathbf{ZD} - \mathbf{ZL}$ to each side of the equation, yielding

$$\begin{aligned}\mathbf{ZD} &= \mathbf{L}'^{-1} + \mathbf{ZD} - \mathbf{ZL} \\ &= \mathbf{L}'^{-1} + \mathbf{Z}\left(\mathbf{D} - \mathbf{L}\right).\end{aligned}$$

So finally, the Takahashi equation for $\mathbf{Z}$ becomes

$$\mathbf{Z} = \mathbf{L}'^{-1}\mathbf{D}^{-1} + \mathbf{Z}\left(\mathbf{D} - \mathbf{L}\right)\mathbf{D}^{-1}.$$

The first term of this equation is an upper triangular matrix because the inverse of an upper triangular matrix is also upper triangular. The diagonal elements of this matrix are the squares of the reciprocals of the diagonal elements of $\mathbf{L}$, which will be denoted as $d_{i,i}$. The second term of this equation is a multiplication of a dense matrix, $\mathbf{Z}$, and a strictly lower triangular matrix with zeroes on the diagonal, $\left(\mathbf{D} - \mathbf{L}\right)\mathbf{D}^{-1}$.

Because **A** is a symmetric matrix, **Z** is also symmetric, and thus we only have to calculate the lower triangular part of **Z**. The diagonal elements are then calculated as

$$z_{i,i} = \frac{1}{d_{i,i}^2} - \frac{1}{d_{i,i}} \sum_{k=i+1}^{n} z_{k,i} l_{k,i}$$

and the off-diagonal elements, for $j < i$, as

$$z_{i,j} = -\frac{1}{d_{i,i}} \sum_{k=j+1}^{n} z_{k,i} l_{k,j} \,,$$

where terms with $l_{k_j} = 0$ can be skipped. These equations can be traversed iteratively with i going down from n to 1 and j from i to 1 and can be used for dense and sparse matrices. However, for a sparse matrix, it is known that for $k > i$

$$l_{i,j} \neq 0, l_{k,j} \neq 0 \Rightarrow l_{k,i} \neq 0$$

and so a subset of elements of **Z** corresponding to non-zero elements in **L** can be computed using only the non-zero elements of **L** and elements of **Z** also present in this subset [105]. In this way it is not necessary to compute the entire inverse of the sparse matrix resulting in a more efficient use of computing resources in terms of memory as well as computing power.

The supernodal multifrontal method for calculating the decomposition of a matrix can also be modified to compute the subset of the inverse based on the Takahashi equations [106]. In this way, level 3 BLAS can be used for inverting the dense supernodes and several supernodes can be processed in parallel. This is also exploited by the PARDISO package, which was one of the reasons why this package has been chosen to be used in our large-scale genomic prediction implementation.

5.4. Application of high performance computing in genomic and genetic prediction

This section presents a brief summary of the use of high performance computing and sparse matrix algebra in a genomic prediction setting, but will also highlight why further research was needed in this area and in what way high performance computing could be introduced even more. It is thus also a bridge to the following chapters, which will explicitly deal with the

improvements high performance computing can offer to the field of genomic prediction.

5.4.1. Before the introduction of genetic markers

As mentioned in the introduction to this chapter, the field of genomic prediction has always made use of supercomputers to overcome the computational burdens of solving the mixed model equations and inverting the relationship matrices. Until the introduction of genetic markers, the focus was mainly on sparse matrix algebra, because the coefficient matrix of the mixed model equations and the covariance matrices were mainly sparse. Although the sparse matrix algebra as introduced in this chapter was not explicitly integrated from the beginning, Henderson already found a method for directly computing the inverse of a sparse numerator relationship matrix in 1976, making it no longer necessary to build and invert the numerator relationship explicitly [45]. This numerator relationship matrix was commonly used as the covariance matrix for the individual genetic effects, so sparse matrix algebra was in this way implicitly applied in genomic prediction. At that time, Henderson's methods for estimating the variance components, which were in principle variations of the ANOVA method, were applied, not yet requiring the inversion of the coefficient matrix [7].

The rise of restricted maximum likelihood for variance component estimation in the beginning of the 1980s changed the origin of the computational burden for genomic prediction, because the coefficient matrix of the mixed model equations had to be inverted to obtain REML estimates of the variance components. Because Henderson's method can only be used for inverting the numerator relationship matrix, because of its specific properties, it could not be used for inverting the coefficient matrix. Furthermore, the coefficient matrix has to be set up for solving the mixed model equations and when using a direct solving routine, the factorisation of this coefficient matrix can also be used to compute the inverse by forward and backward substitution. However, at that time computing resources were a lot more scarce and for certain analyses it was even not possible to keep the entire coefficient matrix in memory, when being stored as a dense matrix. The derivative-free algorithm for obtaining REML estimates of the variance components, conceived in 1986, tried to overcome this by no longer needing the inversion of the coefficient matrix and by sequentially reading and processing the

data [68]. This algorithm already exploited the sparsity of the coefficient matrix, which was a novelty in variance component estimation for genomic prediction.

Another breakthrough for large-scale genomic prediction was the introduction of a direct sparse solver in the expectation-maximisation algorithm for REML estimation of the variance components in 1990 [34]. This made the EM algorithm again competitive with the derivative-free algorithm, which was previously a lot faster due to the fact that before that time dense inversion was used in common implementations of the EM algorithm. Misztal was the driving force behind computational optimisation of the EM algorithm for use in animal breeding and where at first he used a commercial sparse direct solver, his work led to the development of a sparse matrix package FSPAK, which also incorporated the Takahashi equations for finding a subset of the elements of a sparse matrix [69, 107].

As shown in the previous chapter, it is indeed not necessary to compute the entire inverse of the coefficient matrix, as for example in Eq. (4.20) only the trace of $\mathbf{C}^{ZZ}\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}$ is needed for obtaining an estimate of the variance components of the random (mostly genetic) effects. Commonly, the covariance matrix $\mathbf{G}$ has the following form

$$\mathbf{G} = \begin{bmatrix} \gamma_1\mathbf{G}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \gamma_2\mathbf{G}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \gamma_g\mathbf{G}_g \end{bmatrix}$$

and so the derivative of $\mathbf{G}$ with respect to a variance component γ_i, $\dot{\mathbf{G}}_i$ and the inverse of $\mathbf{G}$ are of the form

$$\dot{\mathbf{G}}_i = \begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{G}_i & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix} \quad \mathbf{G}^{-1} = \begin{bmatrix} \frac{1}{\gamma_1}\mathbf{G}_1^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \frac{1}{\gamma_2}\mathbf{G}_2^{-1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \frac{1}{\gamma_g}\mathbf{G}_g^{-1} \end{bmatrix}.$$

Their product is then of the form

$$\dot{\mathbf{G}}_i\mathbf{G}^{-1} = \frac{1}{\gamma_i}\begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{I} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}$$

and

$$\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1} = \frac{1}{\gamma_i^2}\begin{bmatrix} \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{G}_i^{-1} & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \cdots & \mathbf{0} \end{bmatrix}.$$

The coefficient matrix $\mathbf{C}$ of the mixed model equations can also be written as

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}_1 & \cdots & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z}_g \\ \mathbf{Z}_1'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}_1'\mathbf{R}^{-1}\mathbf{Z}_1 + \frac{1}{\gamma_1}\mathbf{G}_1^{-1} & \cdots & \mathbf{Z}_1'\mathbf{R}^{-1}\mathbf{Z}_g \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Z}_g'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}_g'\mathbf{R}^{-1}\mathbf{Z}_1 & \cdots & \mathbf{Z}_g'\mathbf{R}^{-1}\mathbf{Z}_g + \frac{1}{\gamma_g}\mathbf{G}_g^{-1} \end{bmatrix},$$

leading to the observation that the non-zero elements in $\mathbf{G}^{-1}\dot{\mathbf{G}}_i\mathbf{G}^{-1}$ are also non-zero in $\mathbf{C}_{ZZ}$. If the inverse of $\mathbf{C}$ is defined as

$$\mathbf{C}^{-1} = \begin{bmatrix} \mathbf{C}^{XX} & \mathbf{C}^{XZ_1} & \cdots & \mathbf{C}^{XZ_g} \\ \mathbf{C}^{Z_1X} & \mathbf{C}^{Z_1Z_1} & \cdots & \mathbf{C}^{Z_1Z_g} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{C}^{Z_gX} & \mathbf{C}^{Z_gZ_1} & \cdots & \mathbf{C}^{Z_gZ_g} \end{bmatrix},$$

then the trace that originally needed to be calculated can be reduced to the trace of $\mathbf{C}^{Z_iZ_i}\mathbf{G}_i^{-1}$. To calculate that trace, we only need the diagonal elements of the product and as such only the elements of $\mathbf{C}^{Z_iZ_i}$ that are non-zero in $\mathbf{G}_i^{-1}$ have to be calculated. Because of the fact that the Takahashi equations can return the elements of the inverse matrix corresponding to the non-zero elements in the Cholesky factors, this subset of inverse elements

also contains the elements corresponding to non-zero elements in the original matrix. In this way, when the coefficient matrix is sparse, the traces in the score functions of the REML estimation can be rapidly evaluated using the Takahashi equations.

Due to these highly efficient computational improvements, the software package developed by Misztal is still used today and the BLUPF90 suite of programs for genomic prediction, which is very popular among animal breeders, relies on the FSPAK package [108]. Most of the first results with this package were obtained from utilizing a Cray supercomputer, but a square matrix of rank 8,345, which was only for 0.1% filled with non-zero elements could already be partially inverted on a personal computer in 30 seconds [69]. This family of programs thus greatly increased the usability of genomic prediction in animal breeding farms.

5.4.2. After the introduction of genetic markers

The use of genetic markers in genomic prediction was first introduced at the end of the 1980s, but the number of markers was only limited at that time and so sparse matrix algebra could still be used [12]. However, with the advent of SNP genotyping by the end of the previous millennium, many genetic markers became available, leading to the problem of efficiently incorporating this information in genomic prediction models. The landmark paper of Meuwissen et al. was the first to introduce this dense marker information in genomic prediction by modelling a random effect for each genetic marker instead of modelling a single genetic effect per individual [14]. This dramatically changed the appearance of the mixed model equations as they were no longer sparse but dense and the dimension was equal to the number of genetic markers plus the number of fixed effects, instead of the number of individuals plus the number of fixed effects. Furthermore, the covariance matrix for the genetic marker effects was usually a diagonal matrix, making it easily invertible. Therefore, sparse matrix algebra was of lesser importance for genomic prediction, but due to the constant increase in computing power in those years, the dense coefficient matrices could still be processed when the number of genetic markers remained low (up to 1,000).

By 2008, the number of commercially available genetic SNP markers went up to 10,000 and more and so the processing of the dense mixed model equations

became a time-consuming operation. Legarra and Misztal compared direct solving routines to iterative methods for solving the mixed model equations, where the latter show a much faster performance, but the solving time is linearly dependent on the number of phenotypic records [35]. Therefore, direct solving routines were suggested as a viable alternative when the number of records is a lot higher than the number of SNP markers. At the same time, VanRaden came up with a method to avoid the explicit modelling of genetic marker effects, because dense marker maps became available with about 50,000 SNP markers, making it computationally impractical to process the resulting mixed model equations [16]. Instead of modelling the marker effects explicitly, the genetic marker information was used to set up the covariance matrix between individual genetic effects, reducing the dimensionality of the mixed model equations from the number of genetic markers to the number of individuals genotyped, which resulted in a dimension reduction of about one order of magnitude, leading to the fact that reliabilities could be more easily computed for these models.

Not much later, many efforts were put into the computational side of so-called single-step GBLUP, where phenotypic records of genotyped individuals could be combined with phenotypic records of non-genotyped individuals in a single analysis [36, 109, 110]. Correlations between ungenotyped and genotyped individuals were based on pedigree just as the mutual correlations of the ungenotyped individuals, where mutual correlations between genotyped individuals were based on their genotypes. This leads to a covariance matrix that is partly sparse and partly dense, which is handled by splitting it in a sparse and a dense part and solving the mixed model equations in an iterative way by calculating the matrix-vector products separately for the dense part and the sparse part [36]. It should be noted that the dense part was usually several orders of magnitude smaller than the sparse part of these covariance matrices. However, because the number of genotyped individuals kept on growing, the calculation of the covariance matrix for the genotyped individuals could become time-consuming and, therefore, for the first time, it was investigated in 2011 whether the BLAS and LAPACK routines could aid in decreasing the runtime for setting up this covariance matrix [53]. Parallelisation was also investigated in the same study, but it was limited to shared-memory multiprocessing with OpenMP. In 2014, efforts were made to improve efficiency of the factorisation and inversion of the sparse part for REML estimation of the variance components [39]. Although the supernodal

multifrontal method was introduced in this research instead of the commonly used iterative methods and the implementation made use of the BLAS and LAPACK routines, the implementation did not make use of any form of parallelisation.

Furthermore, all the above computational optimisations were based on the fact that the number of genotyped individuals in the analysis stayed reasonably low (at maximum 10,000) and that the number of SNP markers was a lot larger than this number of genotyped individuals. However, the conclusions of a symposium about the future of processing and analysing large-scale data sets in 2011 summarised that parallel programming paradigms were necessary for utilizing high performance computing infrastructures, because of the continuous growth of genotypic data as well in the number of genotyped individuals as in the number of SNP markers used for genotyping [42]. It is therefore that this research aimed as a first step at introducing distributed computing techniques in the field of genomic prediction for animal breeding. The distributed computing techniques can help to overcome the computational burden of an analysis with many genotyped individuals resulting in a high-dimensional dense coefficient matrix for the mixed model equations.

The field of plant breeding mainly followed the developments made for genomic prediction in animal breeding, but because plants are more influenced by environmental conditions, during the last decade research was oriented more and more towards modelling interactions between the genotype and the environment. For these interaction effects, information is only sparsely available since each observation is performed in a single environment. Moreover, when each marker is modelled to have a different effect in each environment and these marker effects are modelled explicitly, the dimensionality of the mixed model equations can increase dramatically. Therefore, as a second step of this research, the distributed computing methods developed in the first part were coupled with sparse matrix algebra to efficiently use supercomputing power for the analysis of large-scale trial data coming from different environments with the inclusion of genotypic information, resulting in a partly dense, partly sparse system of equations.

PART II

APPLICATIONS IN ANIMAL AND PLANT BREEDING

6 DAIRRy-BLUP: a high-performance computing approach to genomic prediction in animal breeding

6.1. Introduction

In the field of genomic prediction, genotypes of animals or plants are used to predict either phenotypic properties of new crosses or breeding values (EBVs) for detecting superior parents. Since quantitative traits of importance to breeders are mostly regulated by a large number of loci (QTL)[1], high density SNP markers are used to genotype individuals. The most frequently used SNP arrays for cattle consist of 50,000 SNP markers, but even genotypes with 700,000 SNPs are already available [42].

Some widely-used analysis methods rely on a linear mixed model backbone in which the SNP marker effects are modelled as random effects, drawn from a normal distribution [14]. The predictions of the marker effects are known as BLUP, which are linear functions of the response variates. It has been shown that when no major genes contribute to the trait, Bayesian predictions and BLUP result in approximately the same prediction accuracy for the EBVs [21, 22, 23]. At present the number of individuals included in the genomic prediction setting is still an order of magnitude smaller than the number of genetic markers on widely-used SNP arrays, favoring algorithms that directly estimate EBVs, which is in this case computationally more efficient than first estimating the marker effects [16, 28, 36, 40]. Nonetheless, it has been shown theoretically that in order to increase the prediction accuracy of the EBVs for traits with a low heritability, the number of genotyped records should increase dramatically [21]. Most widely used implementations like synbreed [111] and BLUPF90 [108] are limited by the computational resources that can be provided by a single workstation.

The content of this chapter has been published as De Coninck, A., Fostier, J., Maenhout, S. and De Baets, B. (2014) DAIRRy-BLUP: A High-Performance Computing Approach to Genomic Prediction. *Genetics* 197(3):813-822.

[1] E.g. more than 2000 QTLs are already found to contribute to the composition and production of milk in cattle (`http://www.animalgenome.org/cgi-bin/QTLdb/BT/index`)

We present DAIRRy-BLUP, a parallel framework that takes advantage of a distributed-memory compute cluster in order to enable the analysis of large-scale datasets. Additionally, results on simulated data illustrate that the use of such large-scale datasets is warranted as it significantly improves the prediction accuracy of EBVs and marker effects.

DAIRRy-BLUP is optimized for processing large-scale dense datasets with SNP marker data for a large number of individuals ($> 100,000$). Variance components are estimated based on the entire dataset using the average information restricted maximum likelihood (AI-REML) algorithm. Therefore, it distinguishes itself from other implementations of the AI-REML algorithm, which are mostly optimized for sparse settings and not able to make use of the aggregated compute power of a supercomputing cluster [39, 108].

All matrix equations in the algorithms used by DAIRRy-BLUP are solved using a direct Cholesky decomposition. Although efficient iterative techniques exist that can reduce memory requirements and calculation time for solving a matrix equation [35], the choice for a direct solver is motivated by the use of the AI-REML algorithm for variance component estimation on the entire data set. In fact, this Cholesky decomposition can be used multiple times in the algorithm to solve systems of equations with a different left-hand side and it can also be used for calculating the inverse of the coefficient matrix, necessary for AI-REML estimation of the variance components.

6.2. Materials and methods

6.2.1. Simulated data

The simulated data used for benchmarking DAIRRy-BLUP was generated using AlphaDrop [112], which was put forward by the Genetics Society of America (GSA) to provide common datasets for researchers to benchmark their methods [113]. Since no datasets with more than 10,000 phenotypic and genotypic records are yet publicly available, this software was used to create datasets with 10,000, 100,000 and 1,000,000 individuals of a livestock population genotyped for a varying number of SNP sites (9,000, 60,000 and 360,000). AlphaDrop first launches MaCS [114] to create 4,000 haplotypes for each of the 30 chromosomes using an effective population size of 1,000 for the final historical generation, which is then used as the base generation

for the next 10 generations under study. The base individuals of a simulated pedigree had their gametes randomly sampled from the 4,000 haplotypes per chromosome of the sequence simulation. The pedigree comprised 10 generations with 10 dams per sire and 2 offspring per dam, with the number of sires depending on the total number of records that was aimed for (50, 500 or 5,000). Genotypes for individuals from subsequent generations were generated through Mendelian inheritance with a recombination probability of 1% per cM. The phenotypes of the individuals were affected by 9,000 QTL, randomly selected out of the approximately 1,670,000 available segregating sites. The effects of these QTL were sampled from a normal distribution with mean 0 and standard deviation 1. Final phenotypes based on the QTL effects were generated with a user-defined heritability of 0.25. This means that one fourth of the variance of the phenotypes was due to the variance of the sum of the additive QTL effects or the true breeding values (TBV). Boxplots of the phenotypes and of the TBV for a population of 100,000 individuals are shown in Figure 6.1 to illustrate this.

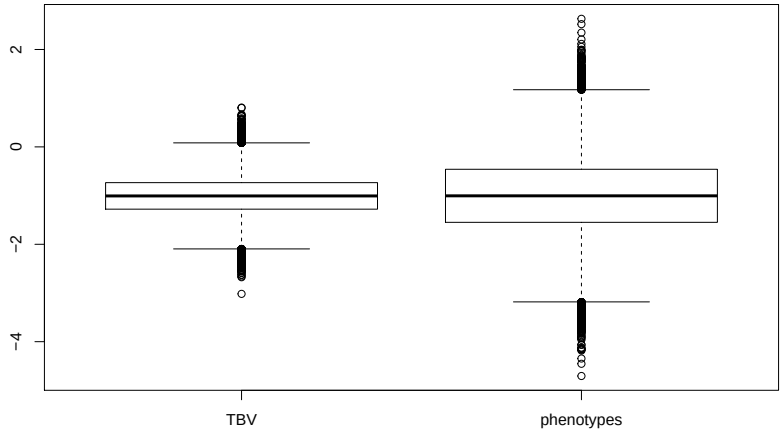

Figure 6.1: A boxplot is given of the simulated true breeding values and the simulated phenotypes for a population of 100,000 individuals.

When constructing the large-scale datasets, it became clear that AlphaDrop, which is a sequential program and does not make use of any parallel programming paradigm, was not designed for generating SNP data for such large numbers of individuals. The largest dataset that could be created

consisted of 1,000,000 individuals, each genotyped for 60,000 SNPs. This required about 118 GB of working memory and the final data set was stored in text-files with a total size of about 350 GB. The simulation took more than 13 days on an Intel Xeon E5-2420 1.90 GHz with 64 GB RAM and another 65 GB as swap memory. The datasets can be recreated by using the correct seeds for the random number generators in the MaCS and AlphaDrop software.

6.2.2. Statistical method: linear mixed model

The underlying framework for the analysis of genotypic datasets is the linear mixed model:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e}\,,$$

where $\mathbf{y}$ is a vector of n observations, $\boldsymbol{\beta}$ is a vector of k fixed effects, $\mathbf{u}$ is a vector of l random effects and $\mathbf{e}$ is the residual error. $\mathbf{X}$ and $\mathbf{Z}$ are the incidence matrices that couple the effects to the observations. Assuming that the random effects and the residual error are normally distributed: $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, \mathbf{G})$ and $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$, the observations are also normally distributed: $\mathbf{y} \sim (\mathbf{X}\boldsymbol{\beta}, \mathbf{V})$, with $\mathbf{V} = \mathbf{R} + \mathbf{Z}\mathbf{G}\mathbf{Z}'$.

The BLUP of the random effects are linear estimates that minimise the mean squared error and exhibit no bias. Henderson translated this estimation procedure into the so-called mixed-model equations (MME), which can be written in matrix form as [8]:

$$\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{Z} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{Z}'\mathbf{R}^{-1}\mathbf{Z} + \mathbf{G}^{-1} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{Z}'\mathbf{R}^{-1}\mathbf{y} \end{bmatrix}$$

These equations were conceived at a time when SNP markers were not yet available and the random effects were associated with a genotypic value per individual. However, when SNP markers were introduced in animal breeding, these equations were used with the SNP markers modeled explicitly and the genotypic value corresponding to the sum of the SNP marker effects [14]. In this study, SNP marker effects are always modeled explicitly to obtain a matrix equation whose dimensionality depends only on the number of SNP markers and the number of fixed effects included in the analysis. As such it is anticipated that genomic datasets will mainly grow in the number of individuals due to the decreasing cost for genotyping [115].

A common simplification, which leads to the ridge regression (RR) formulation [28], is to assume that all SNP marker effects are uncorrelated and have homoscedastic variance σ_u^2. Similarly, residual errors are assumed to be uncorrelated and to have homoscedastic variance σ_e^2. Although these assumptions might not be in accordance with reality, it has been shown that this simplified model can reach prediction accuracies that are close to those of more complex models [16, 40]. The covariance matrices are thus set to $\mathbf{R} = \sigma_e^2 \mathbf{I}_n$ (with $\mathbf{I}_n$ the identity matrix of dimension n) and $\mathbf{G} = \sigma_u^2 \mathbf{I}_l$, which leads to the following MME:

$$\begin{bmatrix} \mathbf{X}'\mathbf{X} & \mathbf{X}'\mathbf{Z} \\ \mathbf{Z}'\mathbf{X} & \mathbf{Z}'\mathbf{Z} + \frac{\sigma_e^2}{\sigma_u^2}\mathbf{I}_s \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{Z}'\mathbf{y} \end{bmatrix}$$

The solution to this matrix equation is known as the Ridge Regression BLUP (RR-BLUP), because $\frac{\sigma_e^2}{\sigma_u^2}$ can be seen as a ridge regression penalization parameter, which controls the penalization by the sum of squares of the marker effects. In this case, however, the penalization parameter is not estimated by cross-validation, but since it is a ratio of two variance components, the latter will be estimated by a Restricted Maximum Likelihood procedure (REML).

In this specific study on simulated datasets, no fixed effects other than the overall mean are modelled, i.e. $k = 1$ and $\mathbf{X} = \mathbf{1}_n$, a vector of n ones. The random effects are the SNP marker effects and every observation corresponds to the phenotypic score of an individual. The $\mathbf{Z}$ matrix is set up using the 0/1/2 coding, where 0 stands for homozygosity in the most frequent allele, 1 stands for heterozygosity and 2 stands for homozygosity in the least frequent allele. An example of the applied linear mixed model with 5 individuals, genotyped for 3 SNP markers is

$$\begin{bmatrix} 1.025 \\ 2.547 \\ -0.015 \\ 0.852 \\ -4.782 \end{bmatrix} = \mu + \begin{bmatrix} 0 & 2 & 0 \\ 1 & 0 & 0 \\ 2 & 0 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{bmatrix} .$$

6.2.3. Average Information REML (AI-REML)

When the covariance structures of the random effects and of the residual errors depend on the respective variance component vectors $\boldsymbol{\gamma}$ and $\boldsymbol{\phi}$,

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{e} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2 \begin{bmatrix} \mathbf{G}(\boldsymbol{\gamma}) & \mathbf{0} \\ \mathbf{0} & \mathbf{R}(\boldsymbol{\phi}) \end{bmatrix}\right)$$

the REML log-likelihood can be written as follows, derived in chapter 3, see Eq. (4.8) [65]:

$$l_{\text{REML}}\left(\sigma^2, \boldsymbol{\gamma}, \boldsymbol{\phi}\right) = -\left((n-k)\log\sigma^2 + \log|\mathbf{G}| + \log|\mathbf{R}| + \log|\mathbf{C}| + \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^2}\right),$$

where $\mathbf{C}$ is the coefficient matrix from the MME as defined in the previous section and

$$\mathbf{P} = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}\left(\mathbf{X}'\mathbf{V}^{-1}\mathbf{X}\right)^{-1}\mathbf{X}'\mathbf{V}^{-1}.$$

With the aforementioned assumptions and setting $\sigma^2 = \sigma_e^2$, $\mathbf{R} = \mathbf{I}_n$ and $\mathbf{G} = \gamma\mathbf{I}_l$, with $\gamma = \frac{\sigma_u^2}{\sigma_e^2}$, this simplifies to a log-likelihood with only 2 parameters, namely σ_e^2 and γ :

$$l_{\text{REML}}\left(\sigma_e^2, \gamma\right) = -\left((n-k)\log\sigma_e^2 + b\log\gamma + \log|\mathbf{C}|\right) + \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma^2},$$

where $\mathbf{C}$ and $\mathbf{P}$ only depend on γ. In this form, we can find an analytical solution for the maximization of the log-likelihood with respect to σ_e^2, provided γ is known:

$$\sigma_e^2 = \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{n-k}.$$

The maximization of the REML log-likelihood function with respect to γ can mostly not be solved analytically, due to the complex first derivative with respect to γ, also known as the score function, which was derived in Chapter 3, Eq. (4.20):

$$\frac{\partial l_{\text{REML}}}{\partial\gamma} = -\left(\frac{b}{\gamma} - \frac{\text{tr}\left(\mathbf{C}^{ZZ}\right)}{\gamma^2} - \frac{\hat{\mathbf{u}}'\hat{\mathbf{u}}}{\sigma_e^2\gamma^2}\right), \tag{6.1}$$

with $\mathbf{C}^{ZZ}$, the lower right block of the inverse of $\mathbf{C}$, also depending on γ. Newton's method for maximizing likelihood functions can be used to find a solution for this optimization problem. The iterative scheme looks like:

$$\boldsymbol{\kappa}_{i+1} = \boldsymbol{\kappa}_i - \mathbf{H}_i^{-1} \nabla l_{\text{REML}} \left(\boldsymbol{\kappa}_i \right) ,$$

with $\boldsymbol{\kappa}_i$ the vector of unknowns (here $\boldsymbol{\kappa}_i = \left(\sigma_e^2, \gamma \right)$) at iteration i, $\mathbf{H}_i$ the Hessian matrix of l_{REML} evaluated at $\boldsymbol{\kappa}_i$ and $\nabla l_{\text{REML}} \left(\boldsymbol{\kappa}_i \right)$ is the gradient vector of the REML log likelihood with respect to $\boldsymbol{\kappa}$, evaluated at $\boldsymbol{\kappa}_i$. However, the Hessian matrix can be very hard to construct due to the large size of the matrices that need to be multiplied or inverted.

Originally, Patterson and Thompson used the Fisher information matrix, which is the expected value of the Hessian matrix, for updating the variance components, but the computation of the elements of this Fisher information matrix involves calculating traces of large matrices, which are tedious to construct [59]. Taking an average of the expected and observed Hessian matrix, these traces disappear in the expression of the update matrix. The resulting update matrix is called the Average Information matrix (AI-matrix) and it can be shown that this AI-matrix can easily be calculated as [65]:

$$\mathbf{H}_{\text{AI}} = \frac{1}{\sigma_e^2} \mathbf{Q}' \mathbf{P} \mathbf{Q} \, , \tag{6.2}$$

with

$$\mathbf{Q} = \begin{bmatrix} \frac{\mathbf{y}}{\sigma_e^2} & \frac{\mathbf{Z}\hat{\mathbf{u}}}{\gamma} \end{bmatrix} . \tag{6.3}$$

The details for obtaining this expression for the average information matrix have been presented in Section 4 of Chapter 3, where it was also shown that this update matrix can be obtained as the Schur complement of $\mathbf{C}$ in an augmented form of this coefficient matrix.

Not only is the AI-REML procedure computationally practical for use in a distributed-memory framework, it has also been shown that less iterations are needed to obtain convergence compared to a derivative-free (DF) algorithm or an expectation-maximization (EM) algorithm [65]. Despite the fact that the computational burden of the AI algorithm is somewhat heavier compared to both DF and EM, the total computation time might be significantly lower due to rapid convergence.

6.2.4. Distributed-memory computing techniques

For large-scale datasets, the application of the RR-BLUP methodology on a single workstation is computationally impractical because of the high memory requirements to store the large and dense coefficient matrix **C**. In order to deal with datasets beyond the memory capacity of a single machine, a parallel, distributed-memory implementation was developed. DAIRRy-BLUP can take advantage of the aggregated compute power and memory capacity of a large number of networked machines by distributing both data and computations over different processes, which can be executed in parallel on the CPU cores of the networked nodes. This allows DAIRRy-BLUP to be executed even on clusters with a low amount of memory per node, because every large matrix in the algorithm is distributed among all parallel processes, i.e. every process holds parts of the matrix and can pass these parts to another process if necessary through message passing [116]. By default the matrix is split up in blocks of 64 by 64 elements, however, this can be changed in function of matrix size and computer architecture. Nonetheless, the matrices are always distributed over the processes using a two-dimensional block cyclic data layout as presented in the previous chapter.

The coefficient matrix **C** has dimensions of $(k + l) \times (k + l)$, the sum of the random and fixed effects, however, in order to construct it, we need to multiply matrices with dimensions l, the number of random effects, by n, the number of individuals, and k, the number of fixed effects, by n. In genomic data sets, l, the number of SNP markers, is usually a lot larger than k and nowadays n is mostly still smaller than l. However, as will be discussed in further sections, the number of observations (n) should increase to provide accurate estimates of the marker effects and the resulting breeding values. It is expected that indeed the number of genotyped individuals will rise in the future, since the cost of genotyping is constantly decreasing and more data of genotyped animals will be gathered in future years. Therefore, the algorithm is implemented in such a way that the number of observations n does not have an impact on memory usage. To obtain this, the **y**-vector and the **Z** and **X** matrices are read from the input files per strip, as was also suggested by Piepho et al. [55]. These strips consist of a certain number of rows, defined by the blocksize and the number of processes involved, more explicitly, for a $p_r \times p_c$ process grid and a square blocksize of $b \times b$, a strip

consists of only bp_c rows. For instance, the lower-right block of the coefficient matrix $\mathbf{C}_{ZZ}$ is then calculated as follows:

$$\mathbf{C}_{ZZ} = \sum_{i=1}^{q} \mathbf{Z}_i' \mathbf{Z}_i + \gamma \mathbf{I}_l \,, \tag{6.4}$$

with q the number of strips needed to read in the entire matrix $\mathbf{Z}$ and $\mathbf{Z_i}$ the strips of the matrix $\mathbf{Z}$.

All mathematical operations that have to be performed on the distributed matrices rely on the standard distributed linear algebra libraries PBLAS [117] and ScaLAPACK [118]. The MME are solved through a Cholesky decomposition of the symmetric coefficient matrix $\mathbf{C}$. The AI-REML algorithm iterates until convergence of γ and since predictions of the marker effects, depending on γ, appear in the score function of the AI-REML algorithm, a Cholesky decomposition of the updated coefficient matrix is required each iteration. ScaLAPACK overwrites the original coefficient matrix by its Cholesky decomposition, implying that $\mathbf{C}$ has to be set up from scratch at the beginning of every iteration. Nonetheless, when sufficient working memory is available and the construction of $\mathbf{C}$ becomes a time-consuming factor, an option can be enabled in DAIRRy-BLUP to store a copy of the coefficient matrix, which can then rapidly be updated with new values of γ and thus renders a complete set-up of $\mathbf{C}$ unnecessary.

For clarity, we will look at an explicit example with a blocksize of 64×64 and a 3×2 process grid and will denote the blockwise form of $\mathbf{Z}$ as

$$\mathbf{Z} = \begin{bmatrix} \mathbf{Z}_{(1,1)} & \mathbf{Z}_{(1,2)} & \cdots & \mathbf{Z}_{(1,s)} \\ \mathbf{Z}_{(2,1)} & \mathbf{Z}_{(2,2)} & \cdots & \mathbf{Z}_{(2,s)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Z}_{(t,1)} & \mathbf{Z}_{(t,2)} & \cdots & \mathbf{Z}_{(t,s)} \end{bmatrix}$$

At first, it should be noted that PBLAS and ScaLAPACK assume that matrices are always stored column-wise, but the process grid is mapped row-wise. However, reading matrices from files is usually performed row-wise and so reading in a matrix row-wise, actually comes down to reading in its transpose column wise. Therefore, the distributed read-in of one strip of $\mathbf{Z}'$ column-wise comes down to

- process $(1, 1)$ reads in $\mathbf{Z}_{(1,1)}$, $\mathbf{Z}_{(1,4)}$, ..., $\mathbf{Z}_{(1,1+i*3)}$,

- process $(2,1)$ reads in $\mathbf{Z}_{(1,2)}$, $\mathbf{Z}_{(1,5)}$, ..., $\mathbf{Z}_{(1,2+i*3)}$,
- process $(3,1)$ reads in $\mathbf{Z}_{(1,3)}$, $\mathbf{Z}_{(1,6)}$, ..., $\mathbf{Z}_{(1,i*3)}$,
- process $(1,2)$ reads in $\mathbf{Z}_{(2,1)}$, $\mathbf{Z}_{(2,4)}$, ..., $\mathbf{Z}_{(2,1+i*3)}$,
- process $(2,2)$ reads in $\mathbf{Z}_{(2,2)}$, $\mathbf{Z}_{(2,5)}$, ..., $\mathbf{Z}_{(2,2+i*3)}$,
- process $(3,2)$ reads in $\mathbf{Z}_{(2,3)}$, $\mathbf{Z}_{(2,6)}$, ..., $\mathbf{Z}_{(2,i*3)}$.

As such, the first strip of $\mathbf{Z}'$, $\mathbf{Z}'_1$ is stored according to the PBLAS standards and $\mathbf{Z}'_1\mathbf{Z}_1$ can be computed and stored in the memory reserved for $\mathbf{C}_{ZZ}$ following Eq. (6.4). The number of strips that have to be read in is here $m/2$, or more generally m/p_c, where m is the upper round of n/b, with n the number of observations and b the blocksize used for splitting up the matrices in blocks.

6.2.5. Algorithmic details: calculation flow

The calculation flow for the DAIRRy-BLUP implementation is presented here. The way how matrices are distributed among processes is not explicitly mentioned since this depends on the number of processes involved and the chosen size of the blocks. The processes are always mapped to a two dimensional grid where the number of columns is as close as possible to the number of rows, since this is recommended in the ScaLAPACK User's Guide [118]. Below are listed the main steps of the algorithm; the complete code can be obtained at `https://github.com/arnedc/DAIRRy-BLUP`.

1. Every process reads in the blocks of the strips of matrices $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{y}$ that are assigned to it by the two-dimensional block cyclic distribution and the coefficient matrix as well as the right-hand side of the MME are calculated using equivalent forms of Eq. 6.4 for obtaining $\mathbf{Z}'\mathbf{Z}$, $\mathbf{X}'\mathbf{X}$, $\mathbf{X}'\mathbf{Z}$, $\mathbf{Z}'\mathbf{y}$ and $\mathbf{X}'\mathbf{y}$. The first two multiplications are calculated using the level 3 PBLAS routine PDSYRK, optimised for computing the symmetric product of a matrix and its transpose, while the others are calculated using level 3 PBLAS routine PDGEMM, optimised for computing the product of two general matrices. All these products are immediately stored as submatrices of $\mathbf{C}$, distributed in a two-dimensional block cyclic way over the local memories of the involved processors. A copy of $\mathbf{Z}'\mathbf{y}$ and $\mathbf{X}'\mathbf{y}$ is kept in memory for use in step 4.

2. Compute the Cholesky decomposition $\mathbf{C} = \mathbf{L}\mathbf{L}'$ in parallel using the PDPOTRF ScaLAPACK routine, optimised for computing the Cholesky decomposition of a symmetric positive definite matrix.

3. Estimates for the random and fixed effects are obtained by directly solving the MME using the Cholesky decomposition of $\mathbf{C}$ and the PDPOTRS ScaLAPACK routine, which performs the forward and backward substitutions

$$\mathbf{L}\mathbf{w} = \begin{bmatrix} \mathbf{X}'\mathbf{y} \\ \mathbf{Z}'\mathbf{y} \end{bmatrix} \qquad \text{and} \qquad \mathbf{L}' \begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\mathbf{u}} \end{bmatrix} = \mathbf{w}\,,$$

where $\mathbf{X}'\mathbf{y}$ and $\mathbf{Z}'\mathbf{y}$ are overwritten by $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{u}}$

4. An estimation for $\sigma_e^2 = \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{n-k}$ can be calculated as the Schur complement of $\mathbf{C}$ in the matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{y}'\mathbf{y} & \mathbf{y}'\mathbf{W} \\ \mathbf{W}'\mathbf{y} & \mathbf{C} \end{bmatrix}, \tag{6.5}$$

because this is equal to $\mathbf{y}'\mathbf{P}\mathbf{y}$ as was shown in chapter 3, with $\mathbf{W} = \begin{bmatrix} \mathbf{X} & \mathbf{Z} \end{bmatrix}$. The estimation of σ_e^2 than comes down to

$$\sigma_e^2 = \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{n-k} = \frac{1}{n-k}\left(\mathbf{y}'\mathbf{y} - \mathbf{y}'\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{y}\right)\,,$$

where $\mathbf{W}'\mathbf{y}$, is the right-hand side of the MME and thus $\mathbf{C}^{-1}\mathbf{W}'\mathbf{y}$ is exactly $\left(\hat{\boldsymbol{\beta}}', \hat{\mathbf{u}}'\right)'$ as calculated in the previous section. The estimation of σ_e^2 thus simplifies to the subtraction of the square of the Euclidean norm of $\mathbf{y}$, which was calculated immediately when $\mathbf{y}$ was read in using the PBLAS routine PDNRM2, and the dot product between the right-hand side of the MME, set-up in step 1, and the vector of estimates $\left(\hat{\boldsymbol{\beta}}', \hat{\mathbf{u}}'\right)'$, which can be calculated by the PBLAS level 1 routine PDDOT.

5. Construct matrix $\mathbf{Q}$ and vectors $\mathbf{X}'\mathbf{Z}\hat{\mathbf{u}}$ and $\mathbf{Z}'\mathbf{Z}\hat{\mathbf{u}}$, by reading in again vector $\mathbf{y}$ and matrices $\mathbf{X}$ and $\mathbf{Z}$ and using level 3 PBLAS routine PDGEMM. Solve the following linear equation with the PDPOTRS

ScaLAPACK routine for vector $\mathbf{v}$:

$$\mathbf{C}\begin{bmatrix}\mathbf{v}_X\\ \mathbf{v}_Z\end{bmatrix} = \frac{1}{\gamma}\begin{bmatrix}\mathbf{X}'\mathbf{Z}\hat{\mathbf{u}}\\ \mathbf{Z}'\mathbf{Z}\hat{\mathbf{u}}\end{bmatrix}, \tag{6.6}$$

using the Cholesky decomposition of $\mathbf{C}$ as computed in step 2.

6. The AI update matrix is calculated as the Schur complement of the coefficient matrix $\mathbf{C}$ inside the matrix M (Eq. (6.5)), with $\mathbf{y}$ replaced by $\mathbf{Q}$ from Eq. (6.3):

$$\mathbf{H}_{\text{AI}} = \frac{1}{2\sigma_e^2}\mathbf{Q}'\mathbf{P}\mathbf{Q} = \frac{1}{2\sigma_e^2}\left(\mathbf{Q}'\mathbf{Q} - \mathbf{Q}'\mathbf{W}\mathbf{C}^{-1}\mathbf{W}'\mathbf{Q}\right). \tag{6.7}$$

Matrix $\mathbf{Q}'\mathbf{Q}$ is square and the diagonal elements can be calculated using the squares of the Euclidian norms of $\frac{\mathbf{Z}\hat{\mathbf{u}}}{\gamma}$ and $\frac{\mathbf{y}}{\sigma_e^2}$, while the off-diagonal elements are the dot product between those two vectors. Furthermore, it should be noted that

$$\begin{aligned}\mathbf{C}^{-1}\mathbf{W}'\mathbf{Q} &= \mathbf{C}^{-1}\begin{bmatrix}\frac{\mathbf{X}'\mathbf{y}}{\sigma_e^2} & \frac{\mathbf{X}'\mathbf{Z}\hat{\mathbf{u}}}{\gamma}\\ \frac{\mathbf{Z}'\mathbf{y}}{\sigma_e^2} & \frac{\mathbf{Z}'\mathbf{Z}\hat{\mathbf{u}}}{\gamma}\end{bmatrix}\\ &= \begin{bmatrix}\frac{\hat{\boldsymbol{\beta}}}{\sigma_e^2} & \mathbf{v}_X\\ \frac{\hat{\mathbf{u}}}{\sigma_e^2} & \mathbf{v}_Z\end{bmatrix}\end{aligned}$$

and so the second term for calculating $\mathbf{H}_{\text{AI}}$ in Eq. (6.7) is the matrix product of 2 rectangular matrices, of which the elements have been calculated before, namely $\mathbf{Q}'\mathbf{W}$, $\mathbf{v}_X$ and $\mathbf{v}_Z$ in step 5 as well as $\hat{\boldsymbol{\beta}}$ and $\hat{\mathbf{u}}$ in step 3.

7. Using the Cholesky decomposition the inverse of $\mathbf{C}$ is calculated with the PDPOTRI ScaLAPACK routine and the trace of $\mathbf{C}^{-1}$ is calculated with the BLACS routine DGSUM2D for obtaining the score function as defined in Eq. (6.1). This routine gathers from all participating processes the elements on the diagonal of $\mathbf{C}^{-1}$ and returns the sum in the root process.

8. As long as the relative updates for σ_u^2 and the REML log-likelihood are not smaller than $\epsilon = 0.01$, repeat from step 1 using an updated value for σ_u^2 and σ_e^2. The iterative cycle is also stopped when the maximum number of iterations is reached (default 20). The values of ϵ and the

maximum number of iterations can be changed by the user to obtain faster convergence or more accurate solutions.

9. Optionally, breeding values can be calculated for a large-scale dataset based on the estimates of the SNP marker effects. Genotypes for the testset are read in by all processes in a distributed way as defined by the two-dimensional block cyclic distribution. To minimize memory requirements, breeding values are calculated in strips, which means that only a few breeding values are calculated in parallel by the level 3 PBLAS routine PDGEMM. The number of breeding values calculated in parallel is defined by the chosen blocksize and the number of dedicated processes, equivalently to the set-up of $\mathbf{C}$ as defined in Eq. (6.4).

6.2.6. Input files

As already mentioned, the data files can require more than 100 GB of disk space when stored as a text file. Due to the distributed read-in of the data files, DAIRRy-BLUP can only cope with binary files, which may require even more space when the data is stored as 64 bit integers or doubles. To overcome this massive hard disk consumption, DAIRRy-BLUP can read in data in the Hierarchical Data Format (HDF5) [119]. The HDF5 file format stores data sets in a file-system like format, providing every data set with a /path/to/resource. Metadata can be added to the data sets in the form of named attributes attached to the path or the data set. It is recommended to use the HDF5-filetype for analyzing data with DAIRRy-BLUP due to 2 major advantages.

First of all, HDF5 offers the possibility to compress data into blocks such that up to 10 times less space is consumed by the final HDF5 file than by storing the data in text files. Secondly, if the size of the compressed blocks is equal to the size of the blocks in which the data is distributed across the memory of all dedicated processes, there is also a significant gain in read-in time of the data. Indeed, every process only needs to decompress several blocks before reading in and doesn't have to go through the whole data set as is the case when using conventional binary files.

6.2.7. High performance computing infrastructure

All results were obtained using the Gengar cluster on Stevin, the high performance computing (HPC) infrastructure of Ghent University. This cluster consists of 194 computing nodes (IBM HS 21 XM blades) interconnected with a 4X DDR Infiniband network (20 Gbit/s). Each node contains a dual socket quad-core Intel Xeon L5420 2.5 GHz CPU (8 cores) with 16 GB RAM. To achieve a high performance a one-to-one mapping of processes to CPU cores was applied and the number of dedicated CPU cores was chosen in such a way that there was a fair balance between wall time and occupation of the Gengar cluster, ensuring that the limit of two GB RAM per core was not exceeded.

6.3. Results

The results illustrate the capability of DAIRRy-BLUP to analyze large-scale data sets in a reasonable amount of time, when sufficient computing power is available. Table 6.1 summarizes the computational demands for the different data sets. It is clear from these results that it becomes computationally impractical to use only a single computing node due to the high amount of RAM required. Implementations that use a genomic relationship matrix to estimate genomic EBVs directly, will in this case require at least 40 GB of RAM when only the upper or lower triangular of the symmetric coefficient matrix is stored [16]. There are machines available with this amount of memory, however, the distributed-memory implementation has the advantage of being able to use as much working memory as available on all networked computers, and computing power can thus easily be increased by adding more machines to the network.

Another advantage of DAIRRy-BLUP is the fact that memory usage depends only on the number of fixed and random effects, and not on the number of individuals included in the analysis. Adding information on more individuals only influences the time required for reading the data from the input files and constructing the coefficient matrix, which is observed to scale linearly with the number of individuals. For data sets where genotypic information is obtained with a certain SNP chip, the required amount of working memory will remain constant regardless of the number of individuals included in

Table 6.1: Computational demands of DAIRRy-BLUP for different numbers of SNP markers included in the analysis for a population of 100,000 individuals.

SNPs (l)	Parallel processes	Runtime (hh:mm:ss)	Number of iterations	RAM per process (MB)	Total RAM (GB)
9,000	16	00:10:56	3	93	1.5
60,000	32	02:54:48	2	608	19.5
360,000	720	17:48:10	2	1,154	831

the analysis. As will be shown further on, prediction accuracy of EBVs increases more significantly when adding more individuals to the analysis than when using denser SNP arrays, at least when the SNP arrays are already sufficiently dense so that the QTL and SNP markers are in linkage disequilibrium [120]. DAIRRy-BLUP can easily be used on the same machine for such data sets, where the number of genotyped individuals is constantly increasing. Therefore, more accurate estimates of EBVs are obtained with minimal effort.

6.3.1. Estimated versus true breeding values

Since breeding values are of utmost importance to breeders, it is of course essential to be able to predict these breeding values correctly. In the data sets, the traits were defined by 9,000 random QTL, whose effects were drawn from a normal distribution. For every animal, the QTL genotype is available and thus the true breeding value (TBV) is nothing else but the sum of the simulated QTL effects. Table 6.2 presents the Pearson correlation between the EBVs and TBVs for different sizes of the population in the data set and different numbers of SNP markers used. More formally, the difference between the vector of TBV and the vector of EBV can be summarised as:

$$\text{TBV} = \mathbf{Zu} \qquad \text{EBV} = \mathbf{Z\hat{u}}\,,$$

with $\mathbf{u}$ the vector of simulated QTL effects and $\hat{\mathbf{u}}$ the vector of estimated QTL effects. Unfortunately, generating a data set of 1,000,000 individuals genotyped for 360,000 SNPs was not feasible, due to limitations of the simulation software AlphaDrop. Although the Stevin computing infrastructure includes a machine with about 120 GB of working memory, it seemed

that this was not sufficient to generate this large data set. Nonetheless, DAIRRy-BLUP would be able to analyze such a data set as only the read-in time would be ten times higher compared to the analysis of the data set consisting of 100,000 individuals genotyped for 360,000 SNPs.

Table 6.2: Pearson correlation between the Estimated Breeding Values (EBVs) and the True Breeding Values (TBVs), for datasets with different numbers of individuals and genotyped for different numbers of SNPs. Column 2 displays the Pearson correlation between TBVs and EBVs, based on the estimated QTL effects (see Table 6.3). The missing value in the lower right corner is due to the impossibility to create such a dataset with AlphaDrop.

Pearson correlation between TBVs and EBVs based on				
	QTL (l)	SNPs (l)		
Individuals (n)	9,000	9,000	60,000	360,000
10,000	0.768	0.643	0.700	0.705
100,000	0.939	0.610	0.809	0.839
1,000,000	0.99	0.612	0.897	X

In the second column of Table 6.2, the results are listed for EBVs based on the QTL genotypes. In this case the random effects are the QTL of which it is known that they influence the phenotypic score by a predetermined effect. In the next paragraph, the estimates of the QTL effects are discussed, but here only the EBVs based on the estimated QTL effects are examined. As expected, it is confirmed that when all loci that contribute to the trait are known and genotypes for these loci are available, EBVs are always more accurate than when relying on genotypes for random SNPs. However, even when the exact positions of the QTL are known, it is observed that predictions of the EBVs are more accurate when the number of individuals is an order of magnitude higher than the number of QTL. The results also indicate that it is more opportune to collect genotypic and phenotypic records of more individuals than to increase the number of SNP markers in the genotypic records, at least when the number of markers is already sufficiently higher than the number of QTL involved. Current large-scale genomic evaluations are performed for at most 40,000 individuals genotyped for around 50,000 SNPs [37, 121], where more information is included using sparse pedigree information of non-genotyped individuals. To improve prediction accuracies of these genomic evaluations, it is shown here that instead of using denser SNP arrays, it is more interesting to increase the number of genotyped

individuals.

In Figure 6.2 the estimated breeding values are plotted in function of the true breeding values for a population of 100,000 individuals. It illustrates the fact that EBV based on 60,000 SNPs are better predicted than based on 9,000 SNPs, but using 360,000 SNPs doesn't add that much to the prediction accuracy. However, when EBV are based on 9,000 QTL, the cloud of points as observed in the other plots, narrows down to a nearly linear correlation between the estimated and true breeding values.

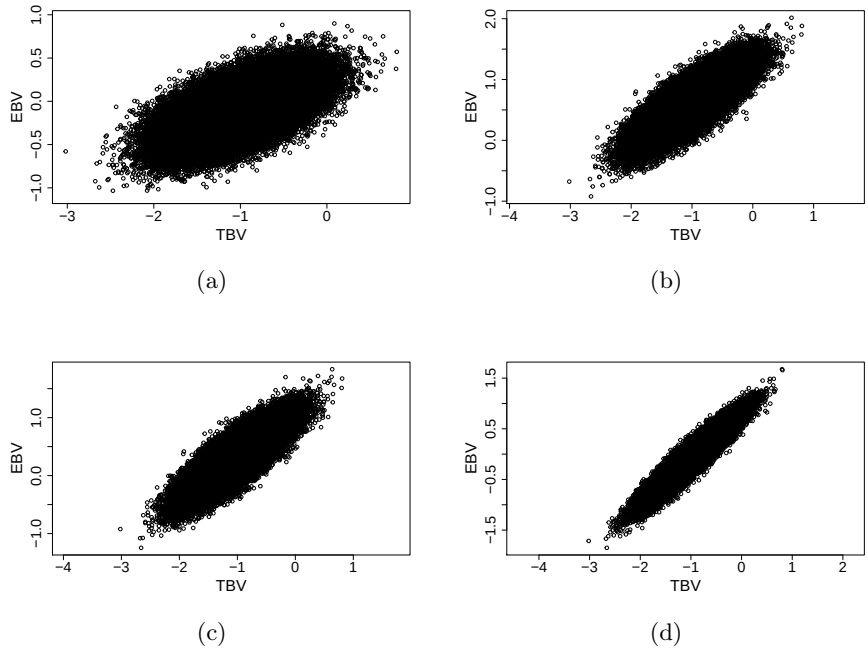

Figure 6.2: Estimated breeding values plotted in function of the true breeding values for a population of 100,000 individuals based on genotypes for (a) 9,000 SNPs, (b) 60,000 SNPs, (c) 360,000 SNPs and (d) 9,000 QTL.

6.3.2. Estimating QTL effects

Breeding values already provide a lot of information to breeders, but it is sometimes interesting to have a good estimate of the marker effects. In the simulated data sets the phenotypic scores are determined by the true QTL effects and the QTL genotypes of the individuals. True SNP marker effects

are not known because SNP markers are randomly drawn and there is no information available about the linkage disequilibrium between the QTL and the SNP markers. Therefore, only the Pearson correlation between the estimated QTL effects and the true QTL effects is given in Table 6.3. When comparing the results of Table 6.3 with those in Table 6.2, it is seen that although the estimates of the marker effects can be quite poor, the resulting estimates of the breeding values might still be fairly accurate. Of course, when trying to identify the positions of important QTL, it is essential to have access to accurate estimates of SNP marker effects and it is observed that large numbers of genotypic and phenotypic records are then required. As an illustration the estimated QTL effects are plotted in function of the simulated QTL effects in Figure 6.3.

Table 6.3: Pearson correlation between the estimated QTL effects and the true QTL effects, for a trait determined by 9,000 QTL

Individuals (n)	Prediction accuracy of QTL effects
10,000	0.354
100,000	0.669
1,000,000	0.851

6.3.3. Parallel efficiency

The primary motivation for the development of DAIRRy-BLUP is that it enables the analysis of large-scale data sets beyond the memory capacity of a single node. This is achieved by the distribution of all matrices and vectors involved in the model among the local memories of a large number of computing nodes. Additionally, DAIRRy-BLUP leads to a significant reduction in runtime because the required computations are performed by several CPU cores concurrently. The parallel speedup $S(P)$ is defined as $S(P) = \frac{T(1)}{T(P)}$, where $T(P)$ denotes the runtime on P CPU cores. The parallel efficiency $\mu(P)$ is defined as the ratio of the parallel speedup and the number of CPU cores used: $\mu(P) = \frac{S(P)}{P}$. In the ideal case, the speedup $S(P)$ is equal to the number of cores used and the parallel efficiency is 100%. However, Amdahl's law states that parallel speedup is limited by the fraction of the program that is parallelizable [80]. Moreover, due to all sorts of overhead such as inter-process communication and load misbalance,

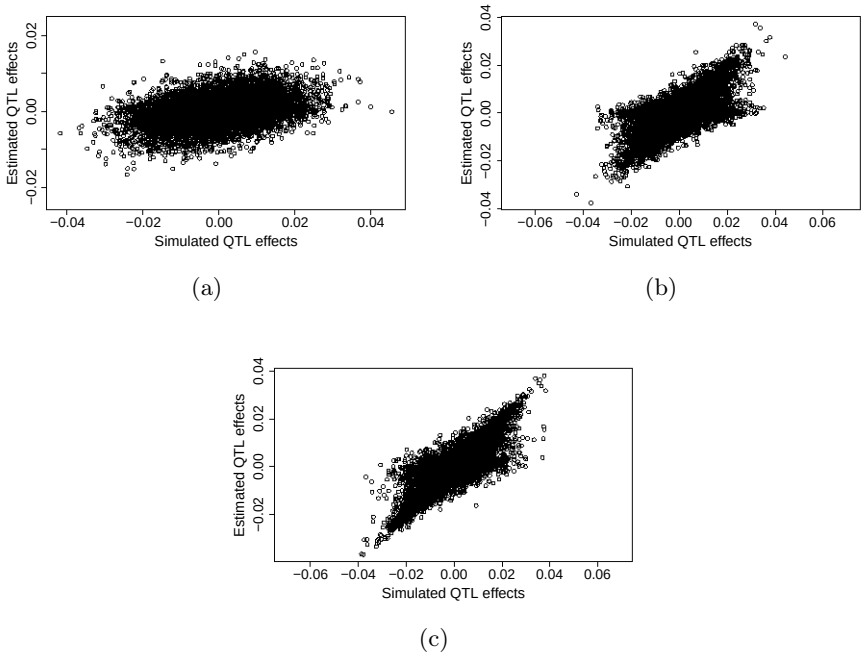

Figure 6.3: Estimated QTL effects are plotted in function of the simulated QTL effects for a population of (a) 10,000, (b) 100,000 individuals and (c) 1,000,000 individuals for a trait determined by 9,000 QTL.

the parallel efficiency is typically even lower than what can be maximally expected from Amdahl's law.

In order to determine the parallel efficiency $\mu(P)$ for the largest problem, i.e., the data set with 100,000 individuals genotyped for 360,000 SNPs, the runtime $T(1)$ on a single CPU core is required. As it is impractical to obtain this value through a direct measurement, it is estimated through extrapolation of the runtime of a smaller problem with 100,000 individuals genotyped for 30,000 SNPs, taking into account the computational complexity of the algorithm. The dominating complexity is determined by the Cholesky decomposition and solution of the MME which are known to have cubic complexity. However, the construction of the coefficient matrix $\mathbf{C}$ is also a time-consuming factor since it has a complexity of $O(nl^2)$ and the number of individuals (n) thus plays an important role when it has the same order of magnitude as the number of SNPs (l), as was the case for both data sets ($n = 100,000$). Therefore, the runtime for the construction of the coefficient matrix was extrapolated separately because of its quadratic complexity in the number of SNPs, as opposed to the cubic complexity of the other time-consuming parts of the algorithm.

Table 6.4 lists the computational demands for the smaller data set with 30,000 SNPs when evaluated on a single CPU core and the estimated runtime and memory requirements for the large data set with 360,000 SNPs on a single CPU core. The timings were averaged over the required iterations, because convergence was reached in a different number of iterations for both data sets. It is clear from this table that the analysis of the 360,000 SNPs data set becomes computationally impractical when not applying any parallel programming paradigm as the processing of the data set would require several months of computing time. By applying DAIRRy-BLUP on a cluster with 720 CPU cores, predictions of marker effects could be calculated roughly 400 times faster. This corresponds to a parallel efficiency in excess of 55%, which is generally considered acceptable.

Table 6.4: Illustration of the parallel efficiency of DAIRRy-BLUP for a dataset with 100,000 individuals genotyped for 360,000 SNPs. Speedup is calculated as the ratio between the computing time on a single CPU core and the computing time on $P = 720$ CPU cores. Parallel efficiency is defined as the ratio between the actual speedup and the maximum speedup, which is equal to the number of dedicated CPU cores.

SNPs (l)	Dedicated CPU cores (P)	Time per iteration	Time for read-in + set-up $\mathbf{C}$	Time for solving MME	Total memory use (GB)
30,000	1	6 h 44 min	5 h 03 min	1 h 41 min	3.73
360,000[a]	1	151 d	30 d	121 d	537
360,000	720	8 h 50 min	1 h 49 min	7 h 00 min	831
Speedup $S(P) = \frac{T(1)}{T(N)}$		416	399	412	-
Efficiency $\mu(P) = \frac{S(P)}{P}$		57.3%	55.4%	57.7%	-

[a] Results for a dataset with 360,000 SNPs on a single CPU core are estimated based on results of the dataset with 30,000 SNPs, cubic complexity of the solving algorithm and square complexity of the set-up of matrix C and memory usage.

Additionally, the memory requirements do not pose any problem, since the distributed-memory framework allows for the aggregation of the local memory (2 GB) available to every CPU core, which means that a total of 1,440 GB RAM could be used.

6.4. Discussion

As has been pointed out by Cole et al., there is a need for being able to process and analyze large-scale datasets using high performance computing methods [42]. DAIRRy-BLUP, a distributed-memory computing framework for genomic prediction is presented and meets some of the issues addressed by Cole et al. It has been shown that DAIRRy-BLUP is able to analyze large-scale genomic datasets in an efficient way when sufficient computing power is available. It is also suggested that analyzing such large-scale genomic datasets is necessary for obtaining better prediction accuracies for marker effects and breeding values, as was already suggested by Hayes et al. and VanRaden et al. [21, 41]. The results show that genotyping more individuals has a stronger effect on prediction accuracy than genotyping for a higher number of SNPs, although an increasing number of SNPs can also provide a higher prediction accuracy. This effect is in accordance with the observations by Habier et al., where a plateau was already reached for around 15,000 SNPs, but the phenotypes were only influenced by 200 QTL as opposed to 9,000 QTL in our study [120]. The number of individuals was also quite low (maximum 2,000), but an increase of prediction accuracy was already noticeable when increasing the number of individuals in the training data. These results, together with the results obtained with DAIRRy-BLUP, justify the choice for an algorithm whose computational demands are dominated by the number of estimated random effects rather than by the number of individuals included in the analysis.

If the exact positions of the QTL on the chromosomes are known and animals can be genotyped for these QTL, prediction accuracies might improve substantially. Additionally, if a large number of phenotypic records of QTL-genotyped individuals is present, the effects of the different QTL can be estimated with significant accuracy. Current real datasets are probably still too small to obtain an accurate estimation of the marker effects; nonetheless, as the cost of genotyping is constantly decreasing, future datasets will contain

the potential for identifying QTL on a large scale.

The results of this study on simulated datasets indicate that for accurate genomic predictions of breeding values, the number of genotyped individuals included in the analysis should increase substantially. Due to the constant decrease in cost for genotyping animals or plants, it is assumed that such large-scale datasets will be available in the near future. Since analysis of these datasets on a single computing node becomes computationally impractical, DAIRRy-BLUP was developed to enable the application of a distributed-memory compute cluster for predicting EBVs based on the entire dataset with a significant speedup.

7 Needles: towards large-scale genomic prediction with marker-by-environment interaction

7.1. Introduction

Genomic prediction methods most often rely on a linear mixed model framework that models at the same time fixed effects as well as random genetic effects [14]. These genetic effects are modeled by assigning a small effect to markers, which are used to genotype the individuals. Introducing a large number of genomewide markers in the analysis has already proven to be beneficial instead of using only pedigree information or a few markers which are known to have a significant effect (so-called marker-assisted selection) in animal breeding [41, 109] as well as in plant breeding [29, 30, 71]. In an animal breeding perspective, environmental effects are mostly not modeled and only regarded as nuisance, because environmental effects are either negligible [122] or can be under the control of the breeders by creating selection environments which are very close to commercial environments [123]. Using this assumption, DAIRRy-BLUP [124] was developed to employ the computing power of supercomputing clusters for analyzing data sets with a large number of genotyped individuals, based solely on dense linear algebra because genetic marker information is mainly dense.

However, when cultivating plants, the environment and some specific environmental conditions (e.g. soil moisture, solar radiation and air humidity) can have a much stronger impact on the phenotypic trait and effects of markers may vary in different environments. It is thus recommended to also include genotype-by-environment interaction effects (G×E effects) for genomic prediction in plant breeding [27, 125]. Different models have been presented to account for these interaction effects in genomic prediction and

The content of this chapter has been submitted as De Coninck, A., De Baets, B., Kourounis, D., Verbosio, F., Schenk, O., Maenhout, S. and Fostier, J. (2015) Needles: towards large-scale genomic prediction with marker-by-environment interaction *Genetics*

most of these models apply a two-stage approach, where in the first stage an adjusted genotype mean is computed across environments, which is then used in the second stage to predict breeding values for untested plants based on their marker genotypes [126]. Actually, this two-stage approach commonly includes a preliminary step in which the intra-environmental effects, such as block, row and column effects are taken into account when computing the genotypic mean per environment. These intra-environmental effects can be modeled together with a location effect and the G×E effects to immediately obtain the genotypic means across the environments in the first step of a two-step approach [126]. However, in recent single-stage analyses, where the computation of genotypic means across environments is avoided and the interaction effects are explicitly modeled, the phenotypic records are mostly already corrected for spatial variations inside the environment [30, 127, 128]. Nonetheless, the single-stage approach may include the modeling of these intra-environmental effects to enable the direct analysis of the raw phenotypic data [31]. The genetic effects can be assumed to follow a wide range of distributions. The most widely-used choice is the assumption that genetic effects come from a normal distribution and while other assumptions may lead to better predictions of the genomic breeding values, the normality assumption is a viable alternative due to its simplicity and computational efficiency [29, 129]. This assumption leads to the so-called best linear unbiased predictors (BLUP) for the random genetic effects [47].

When genetic marker information is applied for calculating correlations between individuals, this is referred to as GBLUP, where the G stands for the usage of a genomic relationship matrix instead of a relationship matrix based on pedigree data [130]. More advanced methods depending on correlations based on pedigree as well as genetic marker information do sometimes result in a slight increase of the prediction accuracy of the breeding values, but the gain in prediction accuracy does mostly not outweigh the added complexity [29, 30]. However, these methods can be of importance when records of ungenotyped individuals should be included in the analysis, which is commonly the case in animal breeding, because historical records of ungenotyped individuals can then be linked to records of genotyped individuals due to the availability of extensive pedigree information [109, 131]. Recently, the GBLUP methodology has been extended to incorporate G×E effects by assuming that the genetic effects were different in each environment, where correlations between genotypes or environments could be included

in the covariance matrices for the random genetic effects and the residual errors [30]. Other methods include a global genetic effect and variable genetic effects across the environments, implying correlation across environments through the shared global genetic effects [128, 132].

In all these studies, genetic marker information is only used to derive more correlations between individuals than when only using pedigree information, while originally in genomic prediction the effect of each marker was modeled explicitly and breeding values were calculated by adding all the genetic marker effects [14]. GBLUP is still preferred as opposed to the explicit modeling of the genetic marker effects, because it enables solving the BLUP equations in the dimensions of the number of genotypic lines used in the study, which is in common data sets still lower than the number of genetic markers included in the linear mixed model. Therefore, the computational burden is still lower for a GBLUP approach than for the explicit modeling of marker effects and their environmental interactions. However, our approach models the genetic marker effects and their environmental interaction effects (M×E effects) explicitly, because we believe that increasing the number of genotypic lines in the analysis has a big potential to increase the prediction accuracy of the model, as was already shown in previous experimental and theoretical research [21, 124]. In such data-driven biology, the number of genotypic lines can in fact become larger than the number of markers, lending support to our explicit modeling approach.

Such explicit modeling of the marker effects was actually first applied in the field of QTL mapping, an important side-track of genomic prediction, helping marker-assisted selection by inclusion of the known QTL as genetic markers [133]. Of course, the field of QTL mapping also includes environmental dependence of the QTL to obtain reliable estimates of the QTL main effects [134]. Linear mixed models used in QTL mapping resemble those of genomic prediction with as a major difference that the QTL effects are mostly modeled as fixed effects next to a random genetic effect, leading to the detection of only a few QTL (max. 40) [135, 136, 137, 138]. However, by treating the QTL effects as fixed effects, they tend to get overestimated, leading to less reliable predictions of the breeding values based on these QTL effects [138]. Therefore, as the explicit modeling of the marker and M×E effects in genomic prediction directly returns predictions for these effects, it seemed interesting to investigate whether these genomic prediction models can be applied for detecting if QTL effects are stable across environments

or not. Although in the next section the statistical model for genomic prediction will be explained generally for SNP markers and their interaction effects, the results section will only be focused on the prediction accuracy of the simulated QTL marker effects and their environmental interaction effects (Q×E effects). In fact, simulating QTL and their effects on the trait is the only way to verify if estimates of genetic marker effect are accurately predicted, because for random SNP markers it can never be ascertained what their exact effects are on the phenotypic trait.

Explicitly modeling M×E effects can lead to huge numbers of effects to be estimated, since every marker is coupled once to each environmental condition. For instance, when plants are tested in 100 different environmental conditions and genotyped for 3,000 markers, this leads to 300,000 M×E effects. Fortunately, the information about the M×E effects is very sparse, because each observation is made under a specific environmental condition, resulting in this case in an incidence matrix for the M×E effects filled for at least 99% with zeroes. Using only dense matrix algebra to provide estimates of all these effects would require huge distributed systems and result in an unnecessary waste of memory and computational resources. Therefore, we exploit the sparsity of the information about the M×E effects and use a technique that couples sparse and dense linear algebra for analyzing such large-scale genomic prediction settings. The first implementation of such a framework, presented hereafter, is called Needles, referring to the saying "finding needles in a haystack", which is very appropriate when trying to find the genetic markers that contribute most to a certain trait out of the results of multi-environment trials. The source code of this implementation can be found on `https://github.com/arnedc/Needles`.

7.2. Materials and methods

7.2.1. Simulated data

All data used for benchmarking were simulated using AlphaMPSim [139], which is flexible simulation software for creating large populations of multi-parent recombinant inbred lines whose polygenic traits are controlled by large numbers of QTL. The default option of AlphaMPSim was chosen to create a historical population of founder haplotypes using MaCS [114]. The

distribution of the AlphaMPSim software included an example for obtaining wheat-like data, mimicking the evolution of wheat according to specified mutation and recombination rates, by employing MaCS. In this way, a pool of 400 haplotypes for each of the 21 chromosomes was provided as a starting point for this study. Actually, wheat is a hexaploid consisting of 3 genomes of 7 chromosomes with homeological effects, but it is commonly modelled as having a diploid genome with 21 chromosomes. A base generation of 2,000 homozygous founders with a genetic diversity as defined by the historical population is then constructed by randomly selecting one of these haplotypes per chromosome. From this base generation 500 lines of 4-way recombinant inbred lines (RILs) were simulated as is schematically depicted in Figure 7.1. The choice for simulating RILs is motivated by the fact that these are used frequently in gene-environment interaction studies and QTL analyses [140, 141]. The pedigree was set up in such a way that there were 50 siblings per line and each of these siblings went through 7 generations of selfing. This resulted in a total number of 203,000 individuals to be simulated, of which the 25,000 of the last generation resemble common wheat RILs.

These 25,000 RIL genotypes were simulated by AlphaMPSim and traits were simulated based on 2 different numbers of QTL (1,575 and 3,150). The global contributions of the QTL to the trait were sampled from a normal distribution with mean zero and standard deviation of one unit. For each genotype, the sum of the global QTL effects is defined here as the true breeding value ignoring any environmental effect and regarding the breeding value as the global genetic potential of a plant. As the goal was to simulate multi-environment trials with different set-ups, random samples were drawn from these 25,000 RIL genotypes and their phenotypes were simulated for different environments. Therefore, 32% of the QTL (resp. 500 and 1,000) were randomly chosen to have a variable effect depending on the environment, sampled from a normal distribution. The total simulated phenotypes were a sum of the true breeding values, the Q×E effects for the given genotype, a fixed environmental effect and a residual term sampled from a normal distribution. This means that no specific row, column or block effects were simulated and thus the phenotypic records can be regarded as already being corrected for spatial variations inside each environment. The variances of the normal distributions were chosen in such a way that of the total variance of the phenotypes, excluding the fixed environmental effects, 37% is attributed

to pure genetic effects, 23% to interactions between the genotype and the environment and the remaining 40% to factors unaccounted for (residual term). These values are in accordance with findings of Jarquín et al. on experimental data about grain yield of 139 wheat lines tested in 340 year × location combinations, which was also the basis for the choice of the variances of the Q×E effects and the residual term [132].

The multi-environment trial was simulated in such a way that in each environment an equal number of different RIL genotypes were tested and 4% of the total number of phenotypic records per environment came from RIL genotypes that were tested across all environments. These so-called standards were replicated 4 times per environment, except for the simulation with the largest number of records where they were replicated 10 times in each environment. As such, the simulation resembles a sort of augmented design [142], where the RIL genotypes that are not standards are tested in 4 different environments, except for the largest simulation where they are tested in all (10) environments. To illustrate this with an example, consider a simulation with 1000 phenotypic records per environment and 50 environments in total. In each environment 40 of the records come from 10 standards that are replicated 4 times in this environment. The other 960 records are simulated from 960 different RIL genotypes. The 10 standards are also replicated 4 times in all the other environments, meaning that of the total number of 50,000 multi-environment trial records, 2000 of these records are obtained by simulating the phenotypic trait of the 10 standards in all environments. The other 48,000 records come from the 960 non-standard RIL genotypes tested in each environment, but because these genotypes are tested in 4 environments, only 12,000 different non-standard genotypes can be included in this multi-environment trial. It must be stressed that an environment in this context is defined as an abstract concept and can be seen as a location, a location × year combination or a class of environments that shared more or less the same environmental conditions.

For a simulation of such a trial, but only on 10 environments, the distribution of the fixed environmental effects, G×E effects, true breeding values (TBV) and the phenotypes is illustrated in Figure 7.2. The distribution of these effects, except for the fixed environmental effect, across the different environments is shown in Figure 7.3. It can be seen that the true breeding values do not vary a lot across the environments as expected and so the variation of the phenotypic records across the environemtns is mainly due

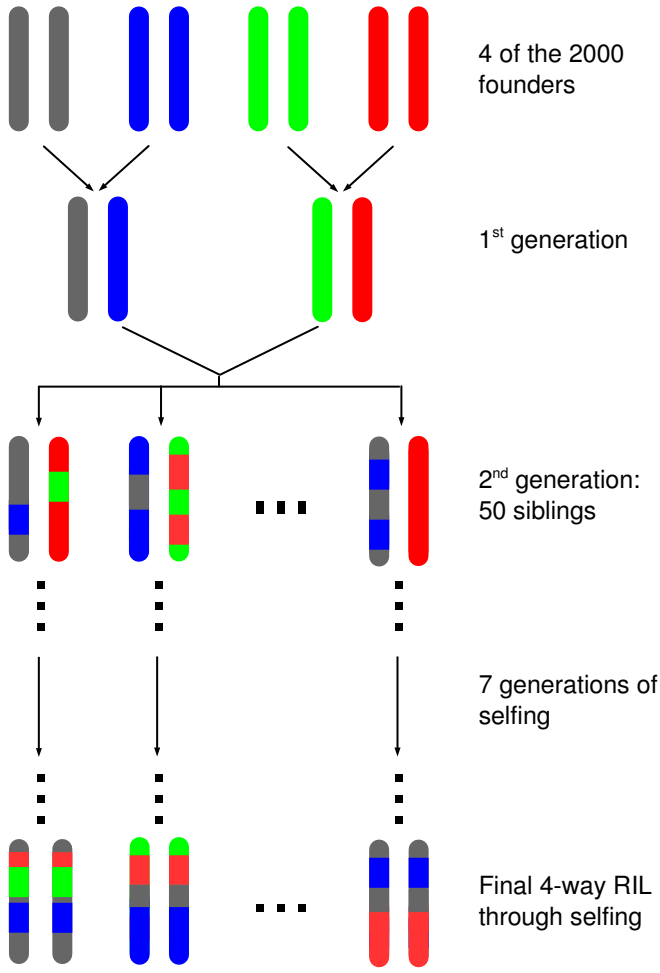

Figure 7.1: Schematic drawing of the set-up of the 4-way recombinant inbred lines. Four founders are used to create an inbred line of which 50 siblings are simulated per line. In total 500 of these lines are created, which results in a pool of 25,000 wheat RIL types that can be tested.

to the fixed environmental effects and the G×E effects.

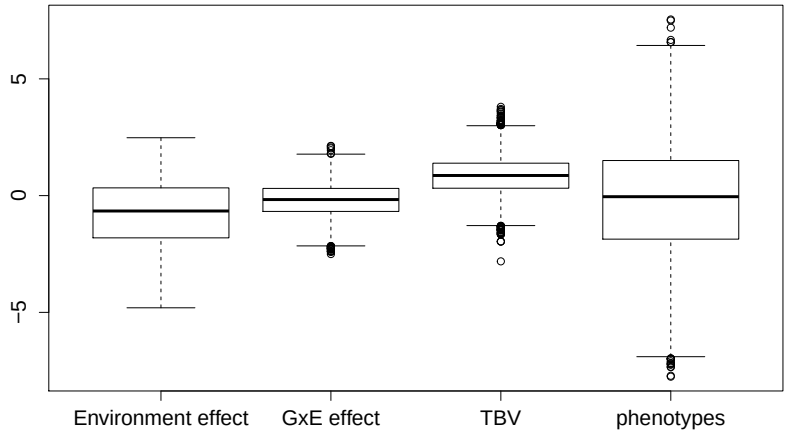

Figure 7.2: Distribution of the simulated phenotypes for a total of 10,000 records coming from 10 different environments.

7.2.2. Statistical method: linear mixed model

The underlying framework of the genomic prediction model with explicit marker-by-environment interaction is the linear mixed model:

$$\mathbf{y} = \mathbf{X}_{\text{env}}\boldsymbol{\beta}_{\text{env}} + \mathbf{Z}_{\text{snp}}\mathbf{u}_{\text{snp}} + \mathbf{Z}_{\text{mei}}\mathbf{u}_{\text{mei}} + \mathbf{e}\,,$$

with $\mathbf{y}$ the vector of n observations, $\boldsymbol{\beta}_{\text{env}}$ the vector of k fixed environmental effects, $\mathbf{u}_{\text{mei}}$ the vector of l random marker-by-environment interaction effects, $\mathbf{u}_{\text{snp}}$ the vector of m random genetic marker effects and $\mathbf{e}$ the residual error term. In the remainder of this chapter, each genetic marker will be assumed to have a variable effect in each environment and thus $l = k \times m$. The design matrices for the effects are respectively $\mathbf{X}_{\text{env}}$, $\mathbf{Z}_{\text{mei}}$ and $\mathbf{Z}_{\text{snp}}$, where the $\mathbf{Z}_{\text{mei}}$ and $\mathbf{Z}_{\text{snp}}$ matrices are $0/1/2$ coded with 0 standing for homozygosity in the most frequent allele, 1 standing for heterozygosity and 2 standing for homozygosity in the least frequent allele. This allele coding was chosen instead of a centered and standardized coding so $\mathbf{Z}_{\text{mei}}$ could be stored more compactly as a sparse matrix. For a specific environment i the model

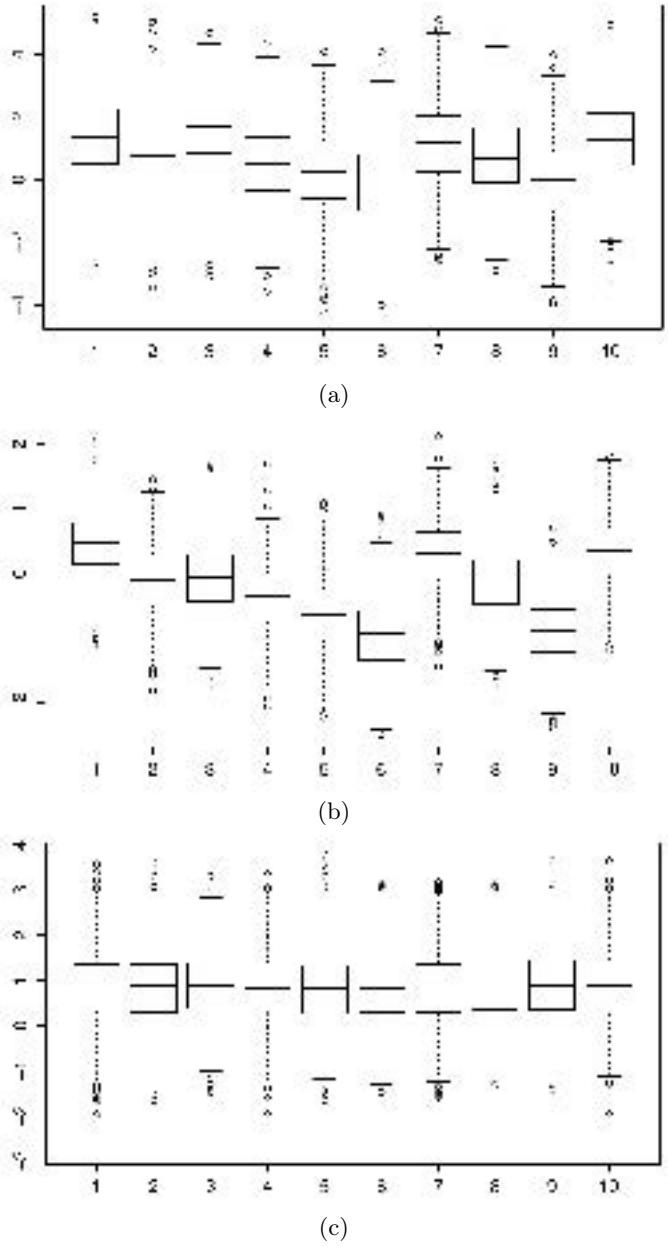

Figure 7.3: Distribution of the different effects contributing to the total phenotype per environment. Boxplot (a) shows the distribution of the phenoytpes corrected for the fixed environmental effect. Boxplot (b) shows the distribution of the G×E effect and boxplot (c) shows the distribution of the true breeding values.

becomes:

$$\mathbf{y}_i = \mathbf{1}\beta_{\text{env},i} + \mathbf{Z}_{\text{snp},i}\left(\mathbf{u}_{\text{snp}} + \mathbf{u}_{\text{mei},i}\right) + \mathbf{e}_i\,,$$

with $\mathbf{1}$ a vector of ones, $\beta_{\text{env},i}$ the fixed environmental effect of environment i, $\mathbf{Z}_{\text{snp},i}$ the matrix of genotypes evaluated in environment i and $\mathbf{u}_{\text{mei},i}$ the marker-by-environment interaction effects in environment i. As the phenotypic records were modeled as being already corrected for spatial variations inside each environment, no specific intra-environmental effects (e.g. block, row or column effects) had to be modeled.

The random effects and residual term are modeled as:

$$\begin{bmatrix}\mathbf{u}_{\text{snp}}\\ \mathbf{u}_{\text{mei}}\\ \mathbf{e}\end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \sigma_e^2 \begin{bmatrix}\sigma_{\text{snp}}^2\mathbf{I}_m & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \sigma_{\text{mei}}^2\mathbf{I}_l & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}_n\end{bmatrix}\right)\,,$$

which results in an a priori assumption that the observations are distributed as:

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{X}_{\text{env}}\boldsymbol{\beta}_{\text{env}}, \mathbf{V}\right),$$

with

$$\mathbf{V} = \sigma_e^2\left(\mathbf{I}_n + \sigma_{\text{snp}}^2\mathbf{Z}_{\text{snp}}\mathbf{Z}'_{\text{snp}} + \sigma_{\text{mei}}^2\mathbf{Z}_{\text{mei}}\mathbf{Z}'_{\text{mei}}\right)\,.$$

Although this model, commonly referred to as a compound symmetry (CS) model, poses some restrictions in the fact that a certain homogeneity between the environments is assumed, it is the most simple method to include marker-by-environment interaction effects and has shown to perform rather well compared to more advanced methods [30]. Moreover, it was shown that this model is the best balance between model complexity and fit to the data when analyzing yield of maize [138]. Because this model required the least number of assumptions about the environmental interaction effects, the data was also simulated using the compound symmetry model.

The Best Linear Unbiased Estimators and Predictors (BLUE and BLUP) of the fixed and random effects are linear estimates or predictions that minimize the mean squared error and exhibit no bias. They are also the

solutions of the so-called Mixed Model Equations (MME) [8]:

$$\begin{bmatrix} \mathbf{X}'_{\text{env}}\mathbf{X}_{\text{env}} & \mathbf{X}'_{\text{env}}\mathbf{Z}_{\text{mei}} & \mathbf{X}'_{\text{env}}\mathbf{Z}_{\text{snp}} \\ \mathbf{Z}'_{\text{mei}}\mathbf{X}_{\text{env}} & \mathbf{Z}'_{\text{mei}}\mathbf{Z}_{\text{mei}} + \frac{\mathbf{I}_l}{\sigma^2_{\text{mei}}} & \mathbf{Z}'_{\text{mei}}\mathbf{Z}_{\text{snp}} \\ \mathbf{Z}'_{\text{snp}}\mathbf{X}_{\text{env}} & \mathbf{Z}'_{\text{snp}}\mathbf{Z}_{\text{mei}} & \mathbf{Z}'_{\text{snp}}\mathbf{Z}_{\text{snp}} + \frac{\mathbf{I}_m}{\sigma^2_{\text{snp}}} \end{bmatrix} \begin{bmatrix} \hat{\boldsymbol{\beta}}_{\text{env}} \\ \hat{\mathbf{u}}_{\text{mei}} \\ \hat{\mathbf{u}}_{\text{snp}} \end{bmatrix} = \begin{bmatrix} \mathbf{X}'_{\text{env}}\mathbf{y} \\ \mathbf{Z}'_{\text{mei}}\mathbf{y} \\ \mathbf{Z}'_{\text{snp}}\mathbf{y} \end{bmatrix}, \tag{7.1}$$

with $\hat{\boldsymbol{\beta}}_{\text{env}}$, $\hat{\mathbf{u}}_{\text{mei}}$ and $\hat{\mathbf{u}}_{\text{snp}}$ the estimators/predictors for $\boldsymbol{\beta}_{\text{env}}$, $\mathbf{u}_{\text{mei}}$ and $\mathbf{u}_{\text{snp}}$. The coefficient matrix of this equation will be referred to as $\mathbf{C}$.

This system of linear equations depends on the variance components σ^2_{snp} and σ^2_{mei}, which are not known a priori. Actually, σ^2_{snp} and σ^2_{mei} are ratios of variance relative to the variance of the residual error σ^2_e, but for ease of notation we will also refer to these variance ratios as variance components. These variance components are estimated using the Average Information Restricted Maximum Likelihood (AI-REML) methodology based on the entire data set. AI-REML is an iterative, gradient-based approach that has shown to converge more rapidly compared to a derivative-free or expectation-maximization approach [65]. The update equation for the variance components is:

$$\boldsymbol{\kappa}_{n+1} = \boldsymbol{\kappa}_n - \mathbf{H}^{-1}_{\mathbf{AI}n} \nabla l_{\text{REML}}(\boldsymbol{\kappa}_n)\,, \tag{7.2}$$

where $\boldsymbol{\kappa}_n$ is the vector of variance components at iteration n, here $\boldsymbol{\kappa} = \left(\sigma^2_e, \sigma^2_{\text{snp}}, \sigma^2_{\text{mei}}\right)'$, $\mathbf{H}_{\mathbf{AI}n}$ is the AI update matrix at iteration n and $\nabla l_{\text{REML}}(\boldsymbol{\kappa}_n)$ is the gradient of the REML log-likelihood with respect to $\boldsymbol{\kappa}$ evaluated at $\boldsymbol{\kappa}_n$. This is very similar to Newton's method for maximizing likelihood functions, but the difference lies in the use of the AI update matrix instead of the Hessian matrix. Originally the expected value of the Hessian matrix, also known as the Fisher information matrix, was used as the update matrix in Eq. (7.2) due to the tedious computation of the Hessian matrix [59]. But the average information matrix, a simplified average of the Hessian and Fisher information matrix, can be constructed in an elegant way and therefore lends itself better to be used in a distributed computing paradigm for updating the variance components each iteration [124].

Using Eq. (4.8) as derived in Chapter 4, with $\mathbf{R} = \mathbf{I}_n$ and

$$\mathbf{G} = \begin{bmatrix} \sigma^2_{\text{snp}}\mathbf{I}_m & \mathbf{0} \\ \mathbf{0} & \sigma^2_{\text{mei}}\mathbf{I}_l \end{bmatrix},$$

the REML log-likelihood function can be written as:

$$l_{\text{REML}}(\sigma_e^2, \sigma_{\text{snp}}^2, \sigma_{\text{mei}}^2) = -\frac{1}{2}\left((n-k)\log\sigma_e^2 + l\log\sigma_{\text{mei}}^2 + m\log\sigma_{\text{snp}}^2 + \log|\mathbf{C}| + \frac{\mathbf{y}'\mathbf{P}\mathbf{y}}{\sigma_e^2}\right),$$

with

$$\mathbf{P} = \mathbf{V}^{-1} - \mathbf{V}^{-1}\mathbf{X}_{\text{env}}\left(\mathbf{X}'_{\text{env}}\ \mathbf{V}^{-1}\mathbf{X}_{\text{env}}\right)^{-1}\mathbf{X}'_{\text{env}}\mathbf{V}^{-1}.$$

When σ_{snp}^2 and σ_{mei}^2 are known, an analytical solution for the maximization of this likelihood function with respect to σ_e^2 can be found. However, for maximization with respect to σ_{snp}^2 and σ_{mei}^2, we have to resort to the iterative AI-REML technique and the gradient of the REML-log likelihood should thus be evaluated for both variance components, based on the results of Chapter 4, Eq. (4.20):

$$\frac{\partial l_{\text{REML}}}{\partial(\sigma_{\text{mei}}^2)} = -\frac{1}{2\sigma_{\text{mei}}^2}\left(l - \frac{\text{tr}(\mathbf{C}^{-1}_{(2,2)})}{\sigma_{\text{mei}}^2} - \frac{\hat{\mathbf{u}}'_{\text{mei}}\hat{\mathbf{u}}_{\text{mei}}}{\sigma_e^2\sigma_{\text{mei}}^2}\right) \tag{7.3}$$

$$\frac{\partial l_{\text{REML}}}{\partial(\sigma_{\text{snp}}^2)} = -\frac{1}{2\sigma_{\text{snp}}^2}\left(m - \frac{\text{tr}(\mathbf{C}^{-1}_{(3,3)})}{\sigma_{\text{snp}}^2} - \frac{\hat{\mathbf{u}}'_{\text{snp}}\hat{\mathbf{u}}_{\text{snp}}}{\sigma_e^2\sigma_{\text{snp}}^2}\right), \tag{7.4}$$

with $\mathbf{C}^{-1}_{(2,2)}$ and $\mathbf{C}^{-1}_{(3,3)}$ the blocks of the inverse of $\mathbf{C}$ corresponding to the blocks in Eq. (7.1) containing resp. $\mathbf{Z}'_{\text{mei}}\mathbf{Z}_{\text{mei}}$ and $\mathbf{Z}'_{\text{snp}}\mathbf{Z}_{\text{snp}}$.

7.2.3. Implementation details

Although the solution of the system in Eq. (7.1) seems trivial, it can become a very computationally demanding task when the number of effects to be estimated becomes so high that the coefficient matrix no longer fits in the working memory of a single computing node. Moreover, the calculation of the inverse matrices in Eq. (7.3) and (7.4) will also become a time-consuming task, since the time complexity is a cubic function of the dimension of the matrix.

To efficiently apply the computing power of a high performance computing cluster, we developed an algorithm that takes advantage of distributed computing techniques and the fact that a large part of the coefficient matrix

$\mathbf{C}$ is filled with zeroes. The latter is due to the fact that every observation comes from a single environment and thus $\mathbf{Z}_{\text{mei}}$ is only sparsely filled with information. Therefore, the coefficient matrix $\mathbf{C}$ is split up in sparse and dense parts:

$$\mathbf{C} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}' & \mathbf{D} \end{bmatrix}, \tag{7.5}$$

with

$$\mathbf{A} = \begin{bmatrix} \mathbf{X}'_{\text{env}}\mathbf{X}_{\text{env}} & \mathbf{X}'_{\text{env}}\mathbf{Z}_{\text{mei}} \\ \mathbf{X}'_{\text{env}}\mathbf{Z}_{\text{mei}} & \mathbf{Z}'_{\text{mei}}\mathbf{Z}_{\text{mei}} + \frac{\mathbf{I}_l}{\sigma^2_{\text{mei}}} \end{bmatrix}, \tag{7.6}$$

$$\mathbf{D} = \mathbf{Z}'_{\text{snp}}\mathbf{Z}_{\text{snp}} + \frac{\mathbf{I}_m}{\sigma^2_{\text{snp}}}, \tag{7.7}$$

$$\mathbf{B}' = \begin{bmatrix} \mathbf{Z}'_{\text{snp}}\mathbf{X}_{\text{env}} & \mathbf{Z}'_{\text{snp}}\mathbf{Z}_{\text{mei}} \end{bmatrix}, \tag{7.8}$$

where $\mathbf{A}$ is a sparse submatrix, while $\mathbf{D}$ and $\mathbf{B}$ are treated as dense submatrices. The solution of the system in Eq. (7.1) as well as the calculation of the two blocks of the inverse of $\mathbf{C}$ can then be performed blockwise. For more information about the blockwise inversion of a matrix, we refer the interested reader to the Appendix of this dissertation.

For the blockwise solution of the system, we will first derive a system of equations equivalent to the system in Eq. (7.1), using the notation as defined above in Eq. (7.5)-(7.8):

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}' & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_D \end{bmatrix} = \begin{bmatrix} \mathbf{y}_A \\ \mathbf{y}_D \end{bmatrix},$$

with

$$\mathbf{x}_A = \begin{bmatrix} \hat{\boldsymbol{\beta}}_{\text{env}} \\ \hat{\mathbf{u}}_{\text{mei}} \end{bmatrix}, \qquad \mathbf{y}_A = \begin{bmatrix} \mathbf{X}'_{\text{env}}\mathbf{y}, \\ \mathbf{Z}'_{\text{mei}}\mathbf{y} \end{bmatrix}$$

$$\mathbf{x}_D = \hat{\mathbf{u}}_{\text{snp}} \quad \text{and} \qquad \mathbf{y}_D = \mathbf{Z}'_{\text{snp}}\mathbf{y},$$

leads to

$$\begin{cases} \mathbf{A}\mathbf{x}_A + \mathbf{B}\mathbf{x}_D = \mathbf{y}_A \\ \mathbf{B}'\mathbf{x}_A + \mathbf{D}\mathbf{x}_D = \mathbf{y}_D \end{cases}$$

Solving the first matrix equation for $\mathbf{x}_A$ and introducing this in the second

equation leads to:

$$\begin{cases} \mathbf{x}_A = \mathbf{A}^{-1}\mathbf{y}_A - \mathbf{A}^{-1}\mathbf{B}\mathbf{x}_D \\ \mathbf{B}'\left(\mathbf{A}^{-1}\mathbf{y}_A - \mathbf{A}^{-1}\mathbf{B}\mathbf{x}_D\right) + \mathbf{D}\mathbf{x}_D = \mathbf{y}_D \end{cases}.$$

Solving the second equation for $\mathbf{x}_D$ leads to:

$$\begin{cases} \mathbf{x}_A = \mathbf{A}^{-1}\mathbf{y}_A - \mathbf{A}^{-1}\mathbf{B}\mathbf{x}_D \\ \left(\mathbf{D} - \mathbf{B}'\mathbf{A}^{-1}\mathbf{B}\right)\mathbf{x}_D = \mathbf{y}_D - \mathbf{B}'\mathbf{A}^{-1}\mathbf{y}_A \end{cases}.$$

This way of solving the mixed model equations involves the construction of the Schur complement $\mathbf{S}$ of $\mathbf{A}$:

$$\mathbf{S} = \mathbf{D} - \mathbf{B}'\mathbf{A}^{-1}\mathbf{B}, \tag{7.9}$$

which transforms the solution of system in Eq. (7.1) into solving respectively:

$$\mathbf{S}\hat{\mathbf{u}}_{\text{snp}} = \mathbf{Z}'_{\text{snp}}\mathbf{y} - \mathbf{B}'\mathbf{A}^{-1}\begin{bmatrix}\mathbf{X}'_{\text{env}}\mathbf{y} \\ \mathbf{Z}'_{\text{mei}}\mathbf{y}\end{bmatrix}$$

$$\mathbf{A}\begin{bmatrix}\hat{\boldsymbol{\beta}}_{\text{env}} \\ \hat{\mathbf{u}}_{\text{mei}}\end{bmatrix} = \begin{bmatrix}\mathbf{X}'_{\text{env}}\mathbf{y} \\ \mathbf{Z}'_{\text{mei}}\mathbf{y}\end{bmatrix} - \mathbf{B}\hat{\mathbf{u}}_{\text{snp}}\,.$$

Moreover, the block $\mathbf{C}^{-1}_{(3,3)}$ of Eq. (7.4) is precisely the inverse of this Schur complement, which is elaborated on in the Appendix. Although only the diagonal elements are needed for calculating the trace of $\mathbf{C}^{-1}_{(3,3)}$, the entire inverse of $\mathbf{S}$ is needed for obtaining these diagonal elements. For a sparse matrix however, a selected inverse can be computed, only requiring the elements of the inverse corresponding with non-zero elements in the factors of the sparse matrix [104]. The diagonal elements of $\mathbf{C}^{-1}_{(2,2)}$ can then be calculated using $\mathbf{S}^{-1}$ and the selected inverse of $\mathbf{A}$:

$$\mathbf{C}^{-1}_{(2,2)_{i,j}} = \mathbf{A}^{-1}_{i,j} + \mathbf{Y}_{i,\cdot}\mathbf{S}^{-1}\mathbf{Y}'_{\cdot,j}\,,$$

with $\mathbf{Y} = \mathbf{A}^{-1}\mathbf{B}$, $\mathbf{Y}_{i,\cdot}$ the i-th row of $\mathbf{Y}$, $\mathbf{Y}_{\cdot,j}$ the j-th column of $\mathbf{Y}'$, $\mathbf{C}^{-1}_{(2,2)_{i,j}}$ element (i,j) of $\mathbf{C}^{-1}_{(2,2)}$ and $\mathbf{A}^{-1}_{i,j}$ element (i,j) of $\mathbf{A}^{-1}$.

The sparse matrix is stored in compressed sparse row format, as it is also used by the PARDISO solver [143, 144, 145], which is applied for solving

the sparse matrix equation and calculating the selected inverse of submatrix **A**. PARDISO is a multi-threaded solver, but it cannot yet cope with a distributed sparse matrix. Therefore, one computing node is entirely reserved for storing sparse submatrix **A** and all CPU cores available on this node are applied for constructing, factorizing and partially inverting it. All other nodes are used for storing dense submatrices **B** and **D** in a one-dimensional block-cyclic column-distributed way. By distributing the matrices in this way, **Y** can be calculated easily by broadcasting **A** to all nodes and solving $\mathbf{AY} = \mathbf{B}$ on each node for all columns of **B** available on this node. As such, the solution **Y** is automatically distributed in a one-dimensional block-cyclic column-distributed way over the available nodes.

7.2.4. High-performance computing infrastructure

All results were obtained using the Delcatty cluster on Stevin, the high-performance computing (HPC) infrastructure of Ghent University. This cluster consists of 160 computing nodes interconnected with an FDR Infiniband network. Each node contains a dual-socket octa-core Intel Xeon Sandy Bridge (E5-2670) 2.5-GHz CPU (16 cores) with 64 GB of physical memory. All data was accessible from storage available through a fast GPFS mount over the Infiniband network. The Delcatty nodes are running Scientific Linux 6.1 (SL6).

The algebraic operations that are to be performed on the distributed dense matrices rely on the standard libraries PBLAS [117] and ScaLAPACK [118] as implemented in Intel MKL 11.2. All communication that is required between the nodes is handled by MPI [116] as implemented in Intel MPI 5.0. The entire code was written in C/C++ and compiled with Intel C++ compiler 15.0. The code makes use of a hybrid MPI/OpenMP parallelism, which means that each MPI process is mapped to an entire node, while OpenMP is used to apply each CPU core on the node to perform the tasks of each MPI process.

Table 7.1: Runtime and memory usage for the analysis of different field trials on 4 computing nodes each equipped with 16 CPU cores. Only a representative subset of all trials analyzed is listed in this table.

Observations	Environmental effects	Genetic marker effects	M×E effects	Iterations upon convergence	Wall time (s)	Average wall time per iteration (s)	Maximum memory per node (GB)
10,000	10	1,575	15,750	5	108	15.6	2.1
100,000	10	1,575	15,750	3	2,480	42.6	2.8
100,000	100	1,575	157,500	5	3,188	163.5	11.2
100,000	10	3,150	31,500	5	9,761	163.4	9.6
100,000	100	3,150	315,000	4	12,650	862.8	43.6
250,000	10	3,150	31,500	6	51,398	339.4	16.1

7.3. Results

The results discussed further on are mainly to show the capabilities of our software and to demonstrate the usefulness of the large-scale analyses that are enabled by this parallel sparse-dense framework. An overview of the required wall time and working memory for different sizes of analyses can be found in Table 7.1. For each analysis, the starting values of the AI-REML algorithm were set to 0.001 and convergence was reached when the relative update of the values of the variance components and the relative update of the log-likelihood were less than 1%. The largest data set analyzed, in terms of number of effects included, resulted in a linear mixed model with 100,000 observations, 100 fixed environmental effects, 3150 random genetic marker effects and 315,000 random M×E effects. The analysis of such a data set could be performed using four computing nodes, with 16 CPU cores per node in 3.25 h. The root node, managing all operations on sparse submatrix **A** required a maximum peak of 40 GB of virtual memory, while the other nodes required a maximum of 44 GB of virtual memory. In total, the analysis thus required 172 GB of memory, while performing the same analysis without any sparse matrix formalism would require more than 800 GB of working memory. It can also be seen in Table 7.1 that for a larger number of observations, the required amount of memory does not change dramatically. Only the computing time increases because the set-up of the coefficient matrix takes more time. This is also reflected in the average wall time per iteration, which does not increase dramatically when more observations are included, because the set-up of the matrices, except for the dense submatrix **D**, is only performed once before the iterative algorithm starts.

A detailed analysis of the time complexity of the algorithm is outside of the scope of this paper as the coupling of sparse and dense matrix formalisms and the several underlying mechanisms of the analysis framework do not allow for a straightforward conclusion. Nonetheless, when all matrices would be treated as dense, the time complexity of the algorithm would be cubic with respect to the number of effects included in the model, as the factorization and inversion of the coefficient matrix **C** dominate the computing time of the algorithm. In Table 7.1 it is clearly seen that the wall time per iteration does not scale cubicly with the number of included effects, which is due to the introduction of the sparse matrix formalism. Factorizing and inverting a

sparse matrix is no longer only a function of the dimension of the matrix but it mainly depends on the number of non-zeroes in the factors of the matrix. As such, it is hard to predict how the wall time per iteration will increase with an increasing number of effects included in the model, but it will always stay below the upper bound of a cubic complexity with respect to the number of effects included in the model.

In the remainder of this paper, marker effects will coincide with QTL effects because simulated data is used and thus the genotypes for the QTL are known together with their true effects. Therefore, the marker-by-environment interaction effects are now referred to as Q×E effects as we actually model the QTL-by-environment interaction effects. Using simulated data makes it possible to evaluate the prediction accuracy of these Q×E effects, because we know the true values of these effects used to simulate the phenotypic records.

7.3.1. Accuracy of estimation of effects

Instead of looking at the accuracy of estimated breeding values or predicted phenotypes, which is commonly done in the field of genomic prediction, we decided to concentrate on the prediction accuracy of the values of the effects. This is made possible by the fact that simulated data is analyzed, while for real-life data the true effects of the genetic markers are not known. The prediction accuracy is measured by the Pearson correlation between the estimated and the true effects, used for simulating the phenotypic traits. Being able to estimate these effects correctly might not only result in a more precise prediction of the phenotypes in certain environments, but also opens up opportunities for pinpointing highly contributing effects and detecting variations of these effects under certain environmental conditions. To obtain some insight into which parameters of a trial set-up influence the prediction accuracy, a range of multi-environment trials were simulated. An overview of the different trial set-ups is given in Table 7.2. Each of these trials was simulated for RILs with a trait determined by 1,575 QTL and one determined by 3,150 QTL, to investigate what the impact of the number of QTL would be on the prediction accuracy.

Table 7.2: Total number of observations in the different simulated trials. Boxes marked with X are not simulated.

Number of environments	RILs per lot[a]					
	20	100	250	500	2500	2500 (10 rotations)
10	800	4,000	10,000	20,000	100,000	250,000
50	4,000	20,000	50,000	100,000	X	X
100	8,000	40,000	100,000	X	X	X

[a] All trials had 4 rotations, unless otherwise mentioned

The prediction accuracy for the fixed environmental effects was always at least 0.97, even for the smallest field trial. The results for the other effects are summarized in Figures 7.4 and 7.5. The estimation of the QTL effects depends mainly on the total number of observations, regardless of the set-up of the field trial (Fig. 7.4). For example, the correlations of the QTL effects for the field trials with in total 100,000 observations for the trait determined by 3,150 QTL are 0.904 (10 environments, 2,500 RILs/lot), 0.902 (50 environments, 500 RILs/lot) and 0.9 (100 environments, 250 RILs/lot), resulting in three overlapping points.

On the contrary, the prediction accuracies of the Q×E effects are mainly affected by the field trial set-up and in particular by the number of observations per environment (Fig. 7.5). Given a fixed number of observations per environment, the prediction accuracies for the Q×E effects are somewhat lower for a lower number of total observations (corresponding to fewer environments), which is probably a side-effect of the fact that the overall QTL effects are also predicted more poorly for these trials. This effect can be seen in Figure 7.5 by the small differences in Pearson correlation for 1000 observations per environment, where the highest correlation is found for a trial on 100 environments (100,000 total observations) and the lowest for a trial on 10 environments (10,000 total observations).

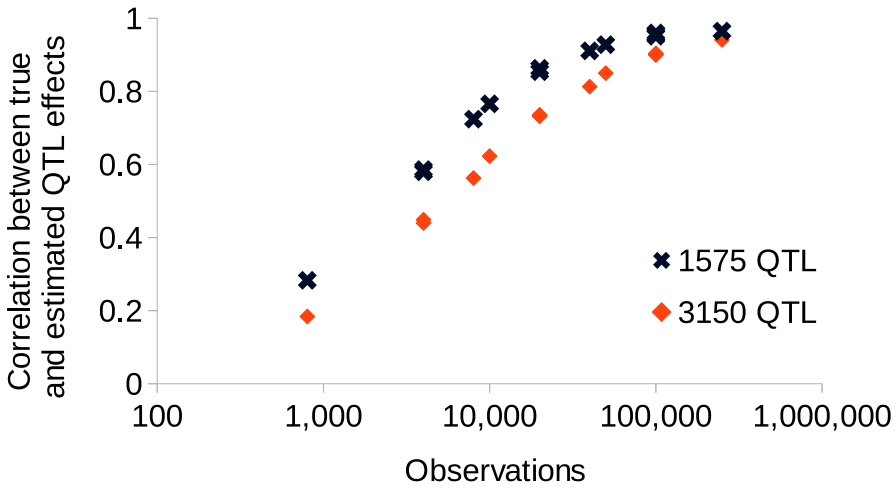

Figure 7.4: Pearson correlation between the estimated QTL marker effects and true QTL marker effects for the different simulated trials in function of the total number of observations.

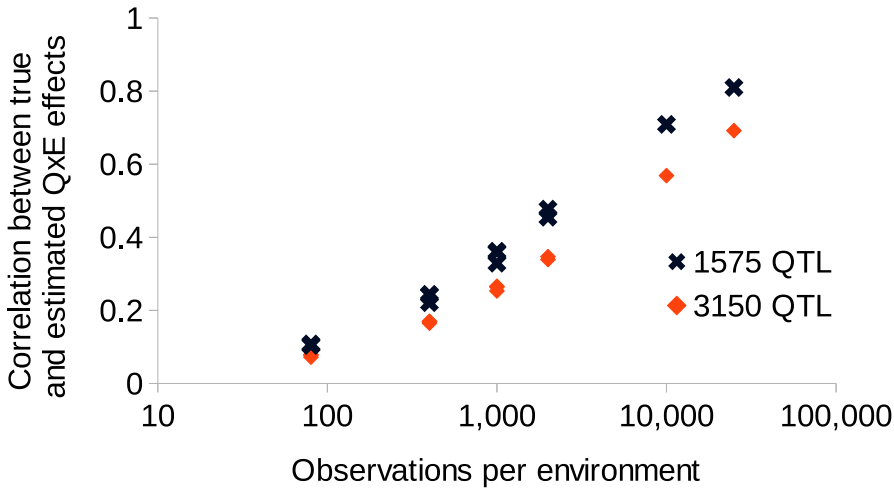

Figure 7.5: Pearson correlation between the estimated Q×E effects and true Q×E effects for the different simulated trials in function of the number of observations per environment.

As already shown theoretically [12], the number of QTL affecting a trait also plays a major role in the predictive power of the genomic prediction model. The more QTL affecting a trait, the bigger the training set should be for accurately predicting the QTL effects (Fig. 7.4). For obtaining a similar prediction accuracy for the Q×E effects when the trait is described by a larger number of QTL, it is again the number of observations per environment that should be increased (Fig. 7.5). To provide a better view on how well these Q×E effects are estimated based on an increasing number of observations per environment, the estimated Q×E effects are plotted in function of the simulated Q×E effects in Figure 7.6. This figure also shows that for low numbers of observations the Q×E effects are mostly ignored by the linear mixed model, but they gain more impact when more observations per environment are present in the data.

7.3.2. Detecting QTL susceptible to environmental conditions

The high prediction accuracies of the Q×E effects for the trials with the most observations per environment triggered the question of how well the model could identify the QTL that contributed the most in certain environments.

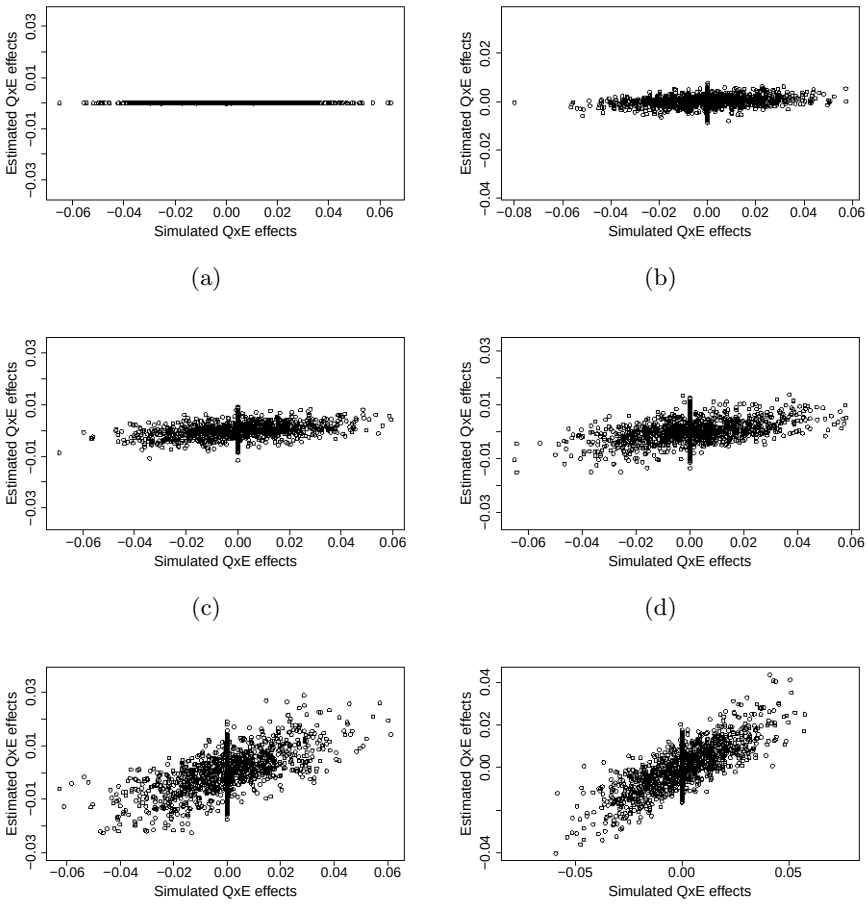

Figure 7.6: The estimated Q×E effects are plotted in function of the simulated Q×E effects for an increasing number of observations per environment: (a) 80, (b) 400, (c) 1000, (d) 2000, (e) 10,000, (f) 25,000

Therefore, the results of the analysis of the trial with 2,500 RILs per lot, undergoing 10 rotations, for a simulated trait with 1575 QTL were examined more thoroughly. The predicted Q×E effects in a certain environment for this trial are plotted in Figure 7.7. From this plot, it can be seen that it is hard to determine how many QTL should be selected as having an important environment-dependent effect. A commonly used approach is to select those with an absolute value greater than a multiple of the standard deviation or interquartile range. However, this always depends on the interpretation of the results and the criterion used may have to change for different traits.

To circumvent these potential problems, a parameter-free methodology akin to machine learning, namely 2-means clustering [146], was used to separate the QTL with a substantial Q×E effect in a specific environment from the other QTL. The usefulness of this technique for different numbers of QTL with simulated Q×E effects was assessed by producing data sets for a trait with 1575 QTL where respectively 5, 25, 50, 100 and 250 of these QTL had environment-dependent effects. After analysis of these data sets and performing 2-means clustering on the estimated Q×E effects, it was seen that the number of QTL in the cluster corresponding with high absolute predicted environmental interaction effects was always less than the number of QTL with a truly simulated Q×E effect. This is in accordance with what is expected as some of these QTL only have a very small simulated Q×E effect in a certain environment and therefore cannot be detected as interacting with that specific environment.

Now that a method is available to select the QTL with a high predicted Q×E effect, some measures are presented to quantify how well this selection corresponds with the underlying truth. The Positive Predictive Value (PPV) is the relative number of QTL found to have important Q×E effects that are also in the set of QTL with a truly simulated Q×E effect. It is thus a measure for the relevance of the detected QTL with an important environmental interaction. The True Positive Rate (TPR) is the relative number of QTL with a truly simulated Q×E effect that are also found by clustering the predicted values for the Q×E effects of the QTL. Therefore, it measures the model's ability to correctly detect QTL that do have important Q×E effects.

Not only is it important to pinpoint the QTL that have a specific environmental effect, the correct estimation of their mutual ranks is desirable to

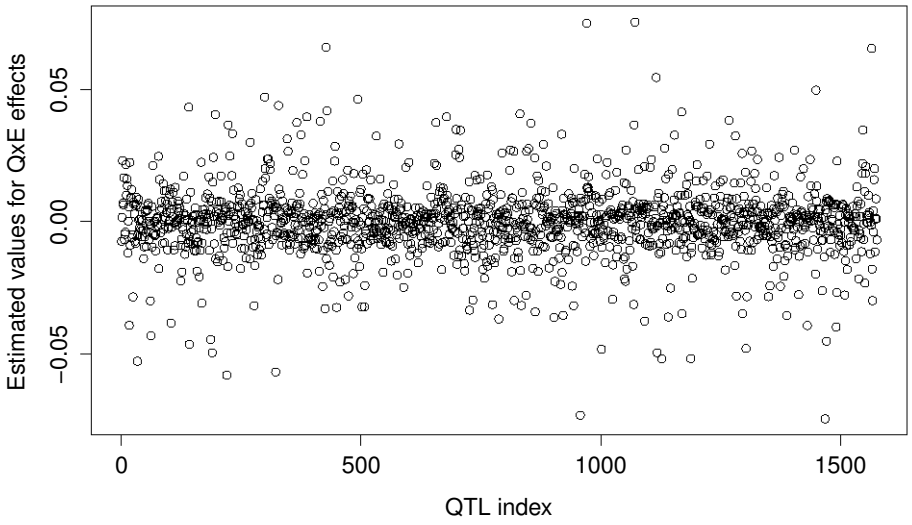

Figure 7.7: Estimated values of the Q×E effects for all included QTL in a certain environment.

enable the selection of the most contributing QTL for a specific environment. A simple measure is the top 10 rank which states how many of the 10 truly most contributing QTL in a certain environment are also in the top 10 of the QTL with the highest predicted Q×E effects. For a more complete measure of the ranking of the QTL, Spearman's rank correlation is used, which is the Pearson correlation of the ranks, without taking into account the actual values of the effects. However, when only a small part of the Q×E effects has a true non-zero value, the Spearman correlation is biased downwards by the fact that the largest part of the true Q×E effects have an equal rank, while this is not the case for the predicted Q×E effects.

Table 7.3: Results of the clustering strategy on the predicted Q×E effects, compared with the raw simulated Q×E effects and the clustering of the simulated Q×E effects. The values in this table are averages over the 10 environments included in the trial, since in each environment the QTL had different effects.

Number of QTL with simulated Q×E effects	Pearson correlation raw[a]	PPV[b] raw	TPR[c] raw	Top 10 rank[d]	Spearman correlation raw	PPV clustered[e]	TPR clustered	Spearman correlation clustered
5	0.725	100%	46.0%	60%[f]	0.671	95.0%	65.3%	0.783
25	0.777	99.4%	42.0%	78%	0.634	90.7%	68.4%	0.777
50	0.801	100%	47.6%	74%	0.686	83.7%	76.0%	0.794
100	0.812	99.5%	42.0%	72%	0.655	84.5%	72.4%	0.780
250	0.796	97.4%	41.3%	54%	0.646	84.5%	68.1%	0.766
500	0.810	95.1%	40.4%	53%	0.659	80.2%	66.1%	0.745

[a] raw: without clustering on the simulated Q×E effects. The clustering of the predicted values is compared with the set of QTL with a truly simulated Q×E effect

[b] PPV: Positive Predictive Value, relative number of QTL in the cluster with the highest absolute predicted Q×E effects that are also in the cluster of QTL with a high simulated Q×E effect

[c] TPR: True Positive Rate, relative number of QTL with a truly high simulated Q×E effect found in the cluster with the highest absolute predicted Q×E effects

[d] Top 10 rank: relative number of the 10 QTL with the highest absolute simulated Q×E effect that are also predicted as being in the top 10 of most contributing QTL in that environment

[e] clustered: with clustering on the simulated Q×E effects. The clustering of the predicted values for the Q×E effects is compared with the cluster of QTL with the highest simulated Q×E effects

[f] Top 5 rank instead of top 10 rank

Therefore, the values of the predicted Q×E effects for the QTL not selected by clustering as having a substantial Q×E effect were set to zero and the Spearman correlation was calculated between the new set of predicted Q×E effects and the simulated Q×E effects in a specific environment. Although this already resulted in a reasonably high rank correlation coefficient, this number still is penalized by comparing the predicted results with an unattainable ideal scenario. To enable a comparison of the results of our model with the best possible results, 2-means clustering was also performed on the simulated Q×E effects. The values of the Q×E effects for QTL belonging to the cluster corresponding with the highest environmental interactions remained unchanged, while the others were set to zero. This corresponds to a scenario where the Q×E effects are estimated correctly and from these estimations the QTL with a high environmental interaction effect are discriminated from the others using 2-means clustering. The Spearman correlation between this set and the new set of predicted values after clustering was calculated, representing a fair measure of how well the most contributing QTL in a certain environment can be ranked.

The results of these analyses are summarized in Table 7.3. The raw Pearson correlation is the same correlation as was used for setting up Figure 7.5 and is a measure of the accuracy of the predicted values for the Q×E effects. However, it is not robust if outliers are present, which can be seen here by the somewhat lower correlation for small numbers of QTL with a simulated Q×E effect. This is also one of the reasons why the Spearman correlation is used as a measure for the accuracy of ranking the QTL in function of their Q×E effects further on. The high numbers of the PPV for the comparison with the ground truth are a very reassuring result, stating that if we can extract QTL that have a substantial Q×E effect, these will almost surely be truly influenced by environmental conditions. In addition, the high top 10 ranks and high PPV for the comparison with the clustered simulated Q×E effects show that those QTL that are found to be have a high Q×E effect are most probably also the QTL with a truly high environment-specific effect. Furthermore, it can be seen that only less than half of the QTL with a simulated Q×E effect can be extracted, but if we compare with what can be extracted in the most ideal situation, cfr. clustering on the simulated Q×E effects, between 65% and 75% are also found by our linear mixed model. Finally, the Spearman correlation between the clustered sets of true and predicted Q×E effects, shows that the model can discriminate

very well between highly contributing, less contributing and negligible QTL in a specific environment. This correlation remains fairly constant when the number of QTL with a simulated Q×E effect is changed, with a minor decrease noted for the highest number of QTL with a simulated Q×E effect.

7.4. Discussion

This paper presented a novel approach of combining sparse and dense matrix algebra together with distributed computing techniques to enable large-scale genomic prediction with marker-by-environment interaction. Due to the increase of genomic data for plants [82], there is a need for a high performance computing framework that can efficiently make use of a supercomputing cluster to analyze these future large-scale data sets. To the best of our knowledge, this is the first attempt of introducing the computing power of a supercomputing cluster in the field of genomic prediction for plants, taking into account variations of genetic effects across environments. It enabled us to analyze data sets with 100,000 observations and more than 300,000 effects in less than four hours, using four computing nodes. Adding more observations increases the runtime due to a more tedious set-up of the system of equations, but this runtime can be decreased by using more computing nodes. The required memory will increase somewhat due to the fact that $\mathbf{X}_{\text{env}}$ and $\mathbf{Z}_{\text{mei}}$ are read in entirely on each node, but due to their sparse storage format this is not a limiting factor for Needles.

As a first result of the analysis of such large-scale data sets with Needles, it has been shown that by increasing the number of phenotyped and genotyped individuals in the training data, the prediction accuracy for the QTL effects can be increased dramatically. Furthermore, if it is desirable to boost the performance in a certain type of environments, it is essential to have as many observations as possible coming from this environment. In this way, the interactions of the QTL with this environment can be estimated fairly accurately and it has been shown that when sufficient observations in this environment are included, the most contributing QTL in this specific environment can be detected by 2-means clustering on the predicted Q×E effects. This result was at first rather unexpected as BLUP are under the influence of shrinkage towards zero that is homogeneous across markers [147],

but apparently it can still identify QTL with a high effect when enough data is available. It thus enables the selection of crops on these QTL, for environment-specific plant breeding.

The selection of environment-dependent QTL by 2-means clustering was chosen because it required no prior knowledge about the Q×E effects. However, because of the fact that the data was simulated by Q×E effects drawn from a normal distribution, and it was also assumed so in our linear model, this information could be included a priori. A more advanced method is known as Gaussian mixture modeling, where clusters are formed by attributing values to being drawn from different normal distributions [54]. In our case, the data is split up into two distributions with a different variance. The results are not shown here, but the main difference with 2-means clustering was the fact that the TPR was somewhat higher, meaning that more QTL with a true Q×E effect could be detected in this way. However, the PPV dropped, making it more unsure that the selected QTL truly had a significant environmental interaction effect.

The post-processing by 2-means clustering of the Q×E effects was proposed because estimations of all possible marker×environment combinations are available as a result of the analysis. The paper by Heslot et al. is the only one we found where M×E effects were also modeled explicitly and where an effort was made to detect QTL with specific environmental variations [127]. However, a Bayesian Lasso approach was used, which made it computationally impractical to model all marker×environment combinations together with the marker effects at once and thus a multi-step approach was needed, leading to quite a complex procedure. Nevertheless, their results indicated that QTL with important Q×E effects had small main effects, which is in contrast with assumptions of earlier studies where Q×E effects were only modeled for QTL with a high main effect [138, 148]. These early studies were mainly interested in mapping QTL that were stable across environments and thus obtain better estimates of the global genetic potential of an individual, regardless of the environment [135, 137]. However, currently attention is directed more towards clarifying which QTL are susceptible to specific environmental covariables such as climatological or soil conditions [127, 138]. This makes it possible to define certain target populations of environments (TPE) sharing some of the same environmental covariables so environmental dependent QTL can be exploited for these TPE [127]. Of course, the weather, playing an important factor in QTL expression [138], is

always partly unpredictable and so information about weather-dependent QTL can open up a whole new research field of balancing between yield optimization for a certain weather type and minimizing the risk of yield loss due to bad weather predictions.

As environments were treated in this study as an abstract concept, they could easily be defined as a TPE with different (simulated) weather conditions to obtain estimates of the Q×E effects under different weather conditions. As such, a certain homogeneity is present between environments, resembling the assumptions made in this paper, and it has been shown that an across-environment model with inclusion of M×E effects then results in better predictions of grain yield for genotypes already evaluated in a certain environment than when using a stratified (i.e. within-environment) analysis [128]. The same study concluded that for untested genotypes or uncorrelated environments results were similar for both types of analyses. However, this could be due to the fact that the number of genotypic lines evaluated per environment was less than 1,000 and as shown here, this might have resulted in bad estimates of the M×E effects, leading to an overestimation of the global marker effects compared to the M×E effects. Moreover, the number of test environments was small (max. 5), so when more environments are introduced, chances are greater that there will always be an environment more correlated with another, and these specific correlations might be introduced in the analytical model via the covariance structure of the residual effects and the M×E effects. The comparison of such an analysis with a stratified analysis might shed more light on the advantages and disadvantages of a joint analysis of multiple environments that are only sparsely correlated and it is thus an interesting path for future research.

The number of observations per environment used in this study might seem high for realistic scenarios, although some studies have shown that the number of observations of different genotypic lines should increase to improve prediction accuracy of complex traits [21, 135]. This also emanates from our simulation study, showing that for the global marker effects 10,000 RILs needed to be evaluated 4 times to surpass a Pearson correlation of 0.8 for a trait determined by 3150 QTL. This number is a lot more than a population of 500 which was suggested by [138] as a limit where beyond there would be probably not much to gain. For improving the prediction accuracy of the Q×E effects it is shown that the number of RILs

evaluated per environment should go well beyond 1,000, what is common practice in realistic scenarios [127, 128]. When environments are treated as location×year combinations, it will indeed be hard to obtain more than 1,000 observations per environment. Nonetheless, similar environments can be clustered together [149] or environments can be categorized by some environmental covariates [132] to increase the number of observations per environment, which might lead to better prediction accuracies of the phenotypic traits in certain types of environments.

All these results were based on analyses where the QTL were known and QTL genotypes were available for all plants. In reality only some of the QTL are known, and therefore genome-wide SNP sets are used for determining the genotype of the plants [82]. In the previous Chapter, it was shown that the prediction of breeding values is a lot harder using SNP markers than when using QTL markers. This can probably be explained by the fact that most SNP markers are only in partial linkage disequilibrium with the QTL, and so it is more difficult to predict their true effects. Future research should clarify if the use of SNP markers instead of QTL genotypes would have a substantial effect on the predictive ability of our model, but it is expected that even more observations are required to detect SNPs that contribute more to a specific trait in a certain environment. Moreover, the statistical model used in this study was intentionally kept simplistic, because the emphasis of this research was to enable the analysis of large-scale data. Using such a simplistic model was made possible by simulating the data according to the model. In more realistic scenarios, this model might not obtain the same performance as the results of this study indicate, but the most important result of this study is the indication that including more genotypic lines in the analysis leads to better prediction accuracies of the genetic effects and their interactions with the environment. A next step to take in the further development of this analytic framework, is the extension of the underlying model so it can take into account distinct genetic variances in different environments and heterogeneous genetic correlations across environments.

Although Needles, a first high performance computing implementation of the approach described in this paper, seems promising for analyzing large-scale genomic data sets obtained from multi-environment trials, there are some side notes to take into consideration. First of all, Needles is optimized for analyzing data sets with a large number of observations (n) in the training data, because this does not influence the memory complexity of

the used model. It is thus not recommended for usage when the number of observations is several orders of magnitude smaller than the number of effects to be estimated. Secondly, an important restriction is the fact that the sparse submatrix cannot be distributed across different nodes. Currently no libraries are available that are able to compute a selected inverse of a distributed sparse matrix, which is also one of the reasons why Needles was developed. When the number of genetic markers and/or the number of environments becomes large, the memory requirements for storing the sparse matrix might surpass the available working memory on a single computing node. Although it was not handled explicitly in this study, there are two ways of dealing with such a situation. As a first step, one could cluster the environments in homogeneous subgroups to reduce the number of different environments [149, 5]. However, when possessing of a large number of genetic markers, the number of marker-by-environment interaction effects may still lead to an unduly large sparse submatrix. Most of the time, not all genetic markers have a variable effect across environments and thus markers can be preselected by screening simultaneously for a high main effect and for consistency of these effects across environments [127, 150]. In this way, one can reduce the number of marker-by-environment interaction effects and thus also the dimensionality of the sparse submatrix $\mathbf{A}$. The number of genetic markers included is then only limited by the characteristics of the computing cluster, because the dense part can be distributed over the working memories of all available computing nodes.

The development of Needles is a first step in taking plant genomic prediction to the next scale. As the cost of genotyping plants decreases and the size of real-life data sets increases, the efficient use of a supercomputing cluster becomes a necessity for enabling the analysis of these data sets. Based on simulated data, it was shown that when a huge amount of training data is available, QTL effects can be predicted accurately and the QTL that have a variable effect in specific environments can be identified. As a further optimization an important next step would be to apply distributed computing techniques on the factorization and selected inversion of the sparse submatrix. This could make it possible to use large-scale SNP genotypes and model marker-by-environment interaction effects for each SNP marker, making it unnecessary to preselect those markers that could have variable effects across environments. We believe that by making the source code open for public, the software can be extended to be used for more complex

models and thus this initial implementation is a stepping stone for future large-scale research in plant breeding.

PART III

EPILOGUE

8 Conclusions and future prospects

This chapter will review the general conclusions that can be drawn from the work in this dissertation. Moreover, because this dissertation is at a crosspoint of two research fields, namely genomic prediction and high performance computing, some future prospects for both areas will be discussed.

8.1. General conclusions

The research presented in this dissertation had as main objective to introduce high performance computing in genomic prediction for animal and plant breeding. The reasons why this research field could benefit from the efficient use of a supercomputing infrastructure were explained in the first part of this dissertation, based on a broad theoretical description of the statistical methodology behind genomic prediction and its history. Historically, especially dairy cattle breeders have always made use of innovative computing methods and facilities to enable the analysis of an ever increasing number of animals. Before the new millennium, computing power increased significantly over the years and so common analysis methods could be used without much adaptation to process ever growing data sets on computers that became more powerful and yet less costly.

However, starting from the landmark paper of Meuwissen et al. in 2001, dense genetic marker maps were introduced in genomic prediction, leading to new computational problems that had to be overcome [14]. This translated the former sparse mixed model equations, used for estimating the genetic merit of an individual, to dense mixed model equations, due to the fact that an effect per genetic marker was estimated. Until 2008, the number of genetic markers used and the number of genotyped individuals were still reasonable, making it unnecessary to dramatically change common analysis methods. But 2008 marked the beginning of the commercial availability of next-generation sequencing technologies, leading to a gigantic decrease in cost for genotyping and the availability of a lot more genetic markers. An illustration of this can be found in Figure 8.1, where it is seen that starting from 2008, the cost of genotyping dropped dramatically. Another indication

of the emerging DNA sequencing technologies are the numerous research articles from the animal and plant breeding research field, dealing with the computational implications of the use of dense marker maps, consisting of many markers [16, 21, 28, 29, 35, 36, 41, 55, 70, 71, 109, 130]. 2008 was also the year that the United States Department of Agriculture officially started using genomic evaluations based on SNP markers for predicting breeding values for milk production of US Holsteins. To give an idea about the increased use of genotypes in these evaluations, in 2009 only 6,508 bulls were genotyped with SNP chips of 50,000 markers, while in 2011 already 116,980 cows and bulls were genotyped with SNP chips of 50,000 markers, but there were also already some genotypes available from SNP chips with 700,000 markers [42, 109]. Currently, around 1 million genotypes are available for US Holsteins, mainly consisting of SNP chips of 50,000 markers, but even whole-genome sequences are readily available for about 121 Holsteins, consisting of 28,336,153 SNPs [151, 152].

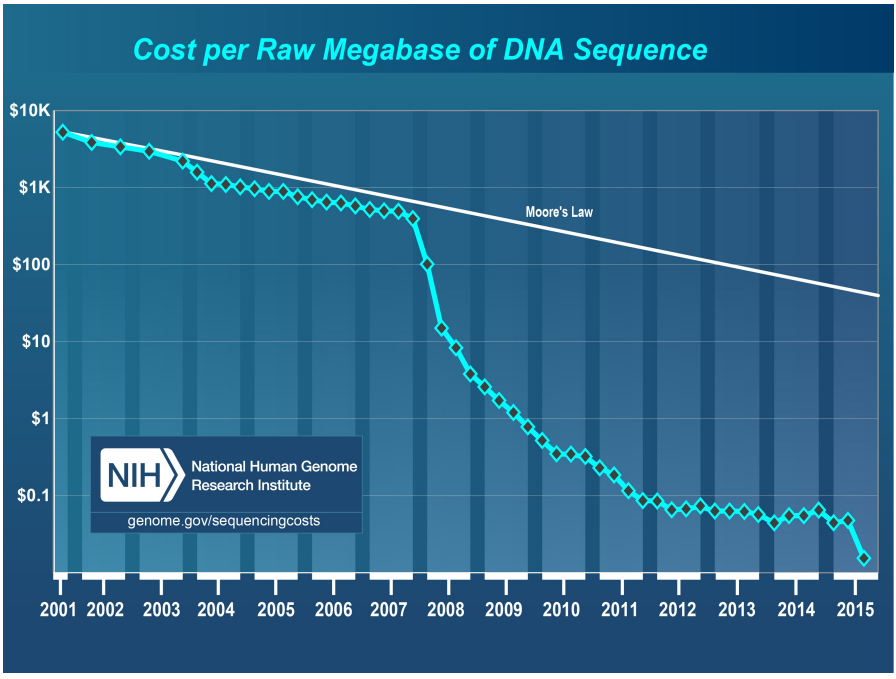

Figure 8.1: Illustration of the evolution of the cost of determining one megabase (Mb; a million bases) of DNA sequence. This cost reflects generating raw, unassembled sequence data. The cost for data analysis, technological developments, informatics equipment and quality assessment is not taken into account. Source: `http://www.genome.gov/sequencingcosts/` [115]

Figure 8.1 represents another important evolution, that is the much faster decrease in cost for genotyping than the decrease in cost for computing power, based on the famous law of Moore [76]. Therefore, it is of vital importance that analysis methods are enhanced in such a way that less computing power is needed for processing data sets of increasing size, or known reliable methods that have proven to perform well are updated so they can make efficient use of supercomputing clusters. The second path has been chosen as the objective for this dissertation, because it can enable processing data sets growing in number of records and/or in number of genetic markers. Moreover, newly developed analysis methods may benefit from the advances made in this research to also exploit high performance computing whenever necessary. This approach had already been suggested by Cole et al. in 2012 as the conclusion of a symposium about the processing and analysis of very large data sets in 2011 [42].

To our knowledge, this dissertation is the first one to investigate the efficient use of a supercomputing cluster for taking genomic prediction to the next level. The first achievement, as discussed in Chapter 6, dealt with a model that is commonly used in animal breeding to predict breeding values of individuals based on the genotype using a dense marker map. Such a model results in a completely dense system of equations with the dimensionality defined by the number of genetic markers. As such, the complexity does not depend on the number of records or genotyped individuals included in the analysis, making it extremely useful for analysing data sets that grow in number of phenotypic records of genotyped individuals with a fixed number of genetic markers on the DNA sequencing chip. But even when the number of markers would rise leading to the inability to process data on the computing platform, the distributed-memory computing method that was applied in DAIRRy-BLUP makes it feasible to extend the computing power and storage capacity of the computing infrastructure by adding more computing nodes to the installed cluster.

The results with this novel implementation confirmed, as already theoretically shown, that increasing the number of phenotypic records of genotyped individuals in genomic prediction, significantly improves the prediction accuracy of the breeding values of the individuals [21, 41]. Increasing the number of SNP markers only improves the prediction accuracy up to a certain level, which was also observed in another study [120]. This is most probably due to the fact that linkage disequilibrium between markers and

QTL is already captured by sampling the DNA at a certain density and going beyond this density does not provide a lot more information. Moreover, it has been shown that effects of markers can be predicted quite accurately if they truly contribute to the trait when a large number of genotyped individuals is available. This was shown by treating the simulated QTL genotypes as marker genotypes and so the results might be poorer for SNP marker genotypes. Nonetheless, using simulated QTL of which the effect on the trait is known is the only way of verifying whether these effects can be accurately predicted.

As a second step, it was tried to resolve a computational burden in plant breeding when interaction effects between the genotype and the environment are included. Usually, these interaction effects are accounted for by calculating an adjusted genotype mean across the environments, which can then be used to predict breeding values in a second stage [31]. However, in this dissertation it was chosen to explicitly model marker effects and their variable effects across the environments (M×E effects). This again leads to mixed model equations whose dimensionality only depends on the number of effects and so including more trial records in the analysis does not lead to computational problems. Moreover, explicitly modelling these M×E effects may also lead to the discovery of QTL that induce different effects under varying environmental conditions. However, the explicit modelling of the M×E effects leads to a dramatic increase in the number of effects to be estimated, because each marker is coupled once to each environment. Nonetheless, information on these effects is only sparsely available, because each observation only comes from a single environment.

The resulting mixed model equations thus are described by a coefficient matrix that is in essence dense, with a large sparse submatrix. This resulted in an implementation that was based upon the first distributed-memory implementation for solving completely dense mixed model equations, extended with the use of a well-performing sparse direct solver, PARDISO [90], to handle the introduction of the large sparse part in the mixed model equations. The results of this combined implementation, obtained using simulated data, showed that QTL effects can again be more accurately predicted when more records of genotyped individuals are included. Moreover, the results indicated that for accurately predicting the variable effects of the QTL in different environments the number of records from different genotypes per environment must be increased. Furthermore, a post-processing based on

2-means clustering was presented to enable the identification of environment-dependent QTL, which performed adequately when the M×E effects were predicted with a high accuracy.

It might seem a hiatus in this thesis that no benchmarking was performed comparing the implemented methods with software packages that are already available and frequently used by animal and plant breeders. Nonetheless, this decision was made with careful considerations in mind. First of all, it should be stressed that the use of high performance computing methods only is beneficial when the data set becomes too large to be analysed on a single computing node. This has as a consequence that for smaller data sets, other software packages will probably outperform our implementation, as some overhead is introduced by enabling the analysis with multiple computing nodes. Secondly, other software packages are to our knowledge limited in the number of records they can process, as they are bound by the RAM memory and number of CPUs available on a single computing node and they can thus not process the large-scale data sets that are discussed in this thesis. Finally, the goal of this thesis was to combine the variance component estimation based on the entire data set combined with the estimation of the different effects. In practice, other software packages reduce the computational requirements by estimating the variance components based on a small data set, which then allows the use of iterative solvers for solving the mixed model equations for obtaining estimates/predictions of the different effects. It would thus be unfair to compare the implementations as presented in this thesis with software packages using small-scale data for which our implementations are not intended or to compare with software packages that do not estimate the variance components based on the entire data set.

8.2. Future prospects in high performance computing

The high performance computing methods used in this dissertation are well established and commonly used in many research fields. The distributed computing methods and in particular the dense matrix algebra based on these distributed computing methods were already conceived in 1992 and have been further optimised later on [87, 89]. Currently, high performance libraries are available, optimised for specific types of processors, developed

by the vendors of these processors. Research is, therefore, more oriented towards fault tolerance and error detection on large-scale clusters [153, 154] and towards increased user-friendliness and ease of the application of the different libraries [155, 156]. For the scalable dense matrix algebra applied in this dissertation, there has not been any blocking factor or a point for improvement that has popped up during the implementation of both analysing algorithms.

However, for handling the sparse matrix algebra and the combination of sparse and dense matrix algebra on a distributed system, there are still some open problems that may deserve some attention in the future. First of all, although there exists a sparse BLAS library, it does not offer a functionality for sparse matrix/sparse matrix multiplication. This has been implemented in the Needles software by means of a naive algorithm, but has shown to perform quite poorly when matrices grow in size. For instance, the multiplication of a sparse matrix of size $315,000 \times 100,000$ with its transpose, leading to a sparse matrix filled for 0.5% with non-zeroes required more than 10 minutes to be calculated. There are surely ways for improving such a sparse matrix-matrix multiplication and it would be practical if it was introduced in the sparse BLAS library as a standard routine. To our knowledge there is only a single initiative in trying to optimize sparse matrix algebra routines, known as Combinatorial BLAS [157]. This library even provides functionalities for distributed sparse matrices, which may be interesting when the sizes of the sparse matrices get even larger. In this dissertation, it was explicitly chosen to not incorporate the Combinatorial BLAS due to two reasons. First, the library still seemed premature for use in our framework and the sparse matrix interface was not compatible with PARDISO, requiring some transformations for the sparse matrices, which could lead to suboptimal use of memory. Secondly, our naive implementation was not the largest bottleneck for the algorithm as the sparse matrix-matrix multiplication only had to be performed once for such large matrices and so it did not seem opportune to invest a lot of time in getting to know the combinatorial BLAS interface.

The largest bottleneck for Needles originated from the multiplication of a sparse matrix with a distributed dense matrix. Again, there is no standard interface provided for such multiplications as the sparse BLAS only provides an interface for the multiplication of a sparse matrix with a dense matrix, both stored on a single computing node. It was attempted to use this sparse

BLAS interface, but eventually this option was discarded as it did not provide the functionalities that were necessary for completing the multiplication. As elaborated on in Chapter 5, we need to be able to define the multiplication of certain blocks of the sparse matrix with the blocks that are stored in the different process memories. This is not possible with the standard sparse BLAS routines and because it was thought to be inefficient to extract the blocks from the sparse matrix, a naive algorithm was written to perform the blockwise sparse matrix block/dense matrix block multiplication. This algorithm was already parallelised for shared memory processors using OpenMP, but it still performed quite poorly for large-scale matrices. For instance, the multiplication of a sparse $315,000 \times 100,000$ matrix filled for about 0.5% with non-zeroes with a distributed dense $100,000 \times 3,150$ matrix needed about 2.5 hours of computation time using 4 nodes, each equipped with 16 CPU cores. Fortunately, also this operation had to be performed only once and the calculation time can be diminished by employing more computing nodes. Nonetheless, it would be interesting to have a standard routine available that can cope with such operations, which might be optimised using a kind of efficient blocking of the non-zero values in the sparse matrix as in the supernodal multifrontal method for the factorisation of the matrix.

Another optimisation for the Needles algorithm would be the possibility to solve a sparse system of equations of which the right-hand side is distributed across the memories of different nodes. Especially for the calculation of the Schur complement of matrix $\mathbf{A}$ as in Eq. (7.9) where the matrix equation $\mathbf{AY} = \mathbf{B}$ should be solved for $\mathbf{Y}$, with $\mathbf{B}$ a dense matrix distributed over the memories of the involved computing nodes, such a routine would be of great help. In the current implementation this is resolved by distributing the dense matrices in a one-dimensional column block cyclic data layout and solving the matrix equations $\mathbf{AY}_i = \mathbf{B}_i$ on the nodes that have the columns $\mathbf{B}_i$ of $\mathbf{B}$ stored in their local memory. Of course, this implies that sparse matrix $\mathbf{A}$ should be broadcast to all involved computing nodes, which can lead to some communication overhead. Nevertheless, this approach has been chosen instead of sending columns of $\mathbf{B}$ to the node that stores $\mathbf{A}$, because the solutions then still had to be sent back to the node that originally stored those columns of $\mathbf{B}$ in its local memory.

There exist sparse direct solvers that can handle distributed right-hand sides in the matrix equations such as MUMPS [91] and SuperLU [93] and

they even have the possibility to distribute the sparse matrix as well over the involved computing nodes. These functions are not yet available in PARDISO [90], but PARDISO was chosen because it supports the efficient computation of a selected inverse of a sparse matrix with the Takahashi equations [104]. To our knowledge there is not yet any software package available that can compute such a selected inverse for a sparse matrix that is distributed across the memories of different computing nodes. Such an implementation would also be of great help for the research conducted in this dissertation as it would allow for the modelling of many more marker-by-environment interaction effects as was also discussed in the previous chapter.

In the current research, all involved matrices were symmetric and positive definite. However, in more general cases, the coefficient matrix of the mixed model equations can become semi-positive definite due to linear dependencies between the equations. For handling singularities in the coefficient matrix due to numerical rounding errors, PARDISO uses diagonal pivoting or Bunch-Kaufman pivoting [158]. But in genomic prediction settings, structural linear dependencies can be introduced in the mixed model equations due to overfitting of the fixed effects. Therefore, it might be useful to detect these linear dependencies prior to solving the mixed model equations. As such, these linear dependencies can be left out of the equations, reducing the dimensionality of the coefficient matrix and converting it to a positive definite matrix. In fact, the current implementation assumes that the coefficient matrix of the mixed model equations is positive definite and non-singular. Therefore, it returns an error message whenever a diagonal element of the Cholesky factors is zero. To overcome this error messages, we can iteratively leave out the equations that lead to a zero diagonal element in the factors. For a large number of fixed effects, such iterations may become very time-consuming and it may lead to a loss of information due to the fact that more equations than the bare minimum are left out than strictly necessary to obtain a positive definite coefficient matrix.

To conclude this section, we would like to suggest an idealistic framework that in our opinion should be the final target in high performance computing. This framework should be an integrated sparse/dense library or collection of libraries that consists of highly efficient basic linear algebra operations (such as the BLAS) and algebraic methods using these basic operations to compute matrix factorisations, solutions of systems of linear equations,

matrix inversions and other routines that are available in LAPACK. The framework should run efficiently on shared-memory processors as well as on distributed systems. The main difference with the current ScaLAPACK implementation and its dependencies is the fact that sparse matrix algebra should be coupled in some way to these existing dense matrix libraries and ideally, the software package should possess of some heuristic method to determine whether to treat a matrix as sparse or dense, or whether to split up a sparse matrix in large dense blocks that are distributed across the memories of the involved computing nodes. It is beyond our scope to assess the feasibility of such an approach, but it seems that current developments, like the PETSc project (Portable, Extensible Toolkit for Scientific Computation), are orienting towards this integrated approach [159]. However, the PETSc project does not provide a routine for (selective) inversion and the user still has to decide whether to treat a matrix as dense or sparse.

8.3. Future prospects in large-scale genomic prediction

The methods proposed in this dissertation allow for genomic prediction based on millions of phenotypic observations from individuals genotyped for up to 360,000 markers when no interaction effects with the environment are modelled and up to 3,150 markers when explicit marker-by-environment interaction effects for 100 environments are modelled. As a first and direct consequence, it was shown that increasing the number of genotyped individuals with phenotypic records in the analysis, dramatically improves the prediction accuracy of the breeding values of the individuals, but also the actual marker effects can be predicted with better accuracy. It was even shown that QTL that are dependent on environmental conditions can be correctly distinguished from those that are stable across environments.

Due to the unavailability of real-life large-scale data sets, all the results in this dissertation are based on simulated data. The advantages of using simulated data is that they offer a great versatility in creating very large-scale populations in different ways, enabling the exploration of scenarios that are not yet available in real-life data sets, and that it is known which genetic markers contribute to the trait under study. A disadvantage of using simulated data is that simulations require certain assumptions, which

may not be entirely in accordance with reality. In our case, data was always simulated using the same assumptions as for the analysing method, probably leading to some overestimation of the prediction accuracies of the breeding values and the different effects. Nonetheless, as a first validation, it is vital to test whether at least in the case of using simulations with the same assumptions as for the analysis, the simulated effects are correctly retrieved by the analytical model. However, now that this first validation has led to satisfactory results, it may be tested to what extent the results of this dissertation hold when real-life data sets are analysed. Unfortunately, real-life data sets with a comparable size as the data sets processed in this dissertation are mostly not publicly available, leading this future path away from the academic world and more towards private institutions and industry.

Of course, another way for further validation of the obtained results could be performed by simulating data using more complex models than those assumed for the analysis. However, the downside of such an approach is that by introducing more complexity, it is not always certain that reality will be better approached. In contrast, keeping a simulation simple in nature usually also does not reflect reality properly, but at least the assumptions are easily verified due to the simplicity of the model. What could be done instead of introducing complexity in the simulation model, is simulating different scenarios that are more practically relevant, perhaps even by using other distributions than the normal distribution for sampling the values of the effects. Some scenarios that could be worth evaluating in dairy cattle breeding are for example:

- Genotype only sires and compare with scenarios where both sires and cows are genotyped.
- Genotype only certain families, and try to predict breeding values of the other families.
- Investigate whether it is still necessary to genotype individuals from a younger generation when genotypes and phenotypes of the forefathers are already available.
- Apply phenotypic or genotypic selection when simulating future generations.
- Combinations of the above.

For all of these scenarios, it can be studied what the effect is of including more records of genotyped individuals. In a plant breeding perspective, where environmental interaction effects are taken into account, there are plenty of scenarios that could be interesting for a specific crop:

- Vary the number of records coming from different environments to make the data sets more unbalanced.
- Apply heterogeneous variances for the simulated Q×E effects in different environments.
- Vary the number of records coming from certain genotypes.
- Apply phenotypic or genotypic selection when simulating future generations.
- Combinations of the above.

Next to validating the approaches used in Chapters 6 and 7, there is of course still room for improvement of the analytical models and implementation of these models. Although the results have shown that including more records of genotyped individuals has a large impact on the prediction accuracy of the different effects, for some analyses it might be that the number of genotyped individuals is an order of magnitude lower than the number of genetic markers. In that case it would be computationally more efficient to employ the GBLUP methodology, where marker information is used to deduce correlations between genotyped individuals and the dimensionality of the mixed model equations is dominated by the number of genotyped individuals. As such, it could be implemented that the algorithm chooses the appropriate methodology based on the characteristics of the data set under study. Moreover, this could open up the path for a combined analysis of genotyped and ungenotyped individuals, where correlations between the two and mutual correlations between ungenotyped individuals are based on the pedigree. This so-called single-step GBLUP (ssGBLUP) has been in use from 2010 and has now even been used for very large-scale genomic predictions [109, 160]. However, although these large-scale genomic predictions involve 570,000 genotyped individuals, only the mutual correlations of a base set of maximum 30,000 animals and the correlations of this base set with the other individuals in this base set is based on the genotypes, while mutual correlations for the individuals not included in the base set are only based on the pedigree information. It might thus be interesting to compare these results with an

approach where all of the correlations are based on the genotypes. Also, the memory requirements for the suggested approach were still quite high, making it perhaps more interesting to apply distributed computing methods.

The models described in Chapters 6 and 7 are admittedly in essence very simple models, because no correlations were assumed between phenotypic values or between any of the random effects, and the number of variance components was low. Nonetheless, such models are commonly used and provide fairly accurate results compared with more complicated models [28, 30, 40]. However, there are several options that could be introduced to enable the analysis of genomic data sets with a greater variety of models. The most straightforward extension would be to introduce the ability to provide matrices $\mathbf{G}$ or $\mathbf{R}$ as input to the program, with $\mathbf{R}$ preferably in sparse format as its dimensionality is defined by the number of observations. This would mainly have an impact on the set-up of the mixed model equations, but also on the variance component estimation as the inverse of $\mathbf{G}$ should be calculated for evaluating the score functions.

The step taken from chapter 6 to chapter 7 in view of the model was actually quite small, as we only introduced one more family of random effects, with a different variance than the other family of random effects. As the focus lies on introducing sparse matrix algebra to the existing distributed-memory framework, the modelling part was kept as simple as possible to avoid introducing undue complexity. However, now that the computational basics are implemented, the road is cleared for exploring more complex models with more random effects. A first improvement could be to enable the modelling of heterogeneous variances for M×E effects for different environments. Next to this extension, more random effects could be modelled, for instance the environmental effects might be modelled more accurately as random effects in stead of as fixed effects. Moreover, a complete single-stage approach could be achieved by enabling the modelling of spatial variations such as block effects, row and column effects and plot effects. Jarquín even suggested to replace the environmental effects by environmental covariates, such as temperature, soil moisture and solar radiation, and model interaction effects between the markers (or genotypes) and the environmental covariates [132]. This would probably lead to mixed model equations that are less sparse, but hopefully high performance computing methods will also evolve so these kinds of mixed model equations could also be processed efficiently on supercomputing clusters.

Another approach could be to cluster certain environments based on the environmental covariates and exploit correlations between the environments based on the environmental covariates. In this case the sparsity of the mixed model equations is preserved and still some conclusions can be drawn about dependency of QTL effects on environmental covariates. Such an integrated approach creates a lot of opportunities in plant and animal breeding. Primarily, it enables the detection of QTL that are stable across environments, which are important to select or create lines that perform well across different environments. However, QTL with a varying effect under certain environmental conditions may be exploited for optimising yield in specific target populations of environments. Such target populations of environments share some similar environmental covariates, which should stay more or less constant. Moreover, some QTL are also subject to changing weather conditions, which may have a huge impact on the yield. Weather conditions are not always predictable and thus they incorporate uncertainty in the genomic prediction for target populations of environments. It thus seems logical that in the future, when more QTL are detected and their dependency on environmental covariates is better known, genomic prediction will have to take into account statistical models for balancing between yield optimisation for a certain weather type and minimising the risk of yield loss due to bad weather predictions. Due to the many effects that can play a role in these risk minimisation methods, it is thought that the results of this dissertation may play a role in the faster development of such analyses.

Appendices

A Results from linear algebra

A.1. Inverse of a square block matrix

The inverse $\mathbf{M}^{-1}$ of a square $n \times n$ matrix $\mathbf{M}$ must satisfy the condition that $\mathbf{M}^{-1}\mathbf{M} = \mathbf{M}\mathbf{M}^{-1} = \mathbf{I}_n$. The columns $\mathbf{M}^{-1}_{\cdot,i}$ of $\mathbf{M}^{-1}$ can thus be found by solving the linear system of equations $\mathbf{M}\mathbf{M}^{-1}_{\cdot,i} = \mathbf{e}_i$, with $\mathbf{e}_i$ a vector with all elements equal to zero except for the ith element, which is one.

When matrix $\mathbf{M}$ and also $\mathbf{M}^{-1}$ are split up in blocks,

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \qquad \mathbf{M}^{-1} = \begin{bmatrix} \mathbf{M}^{(1,1)} & \mathbf{M}^{(1,2)} \\ \mathbf{M}^{(2,1)} & \mathbf{M}^{(2,2)} \end{bmatrix},$$

with $\mathbf{A}$ and $\mathbf{M}^{(1,1)}$ square invertible $n_1 \times n_1$ matrices, $\mathbf{D}$ and $\mathbf{M}^{(2,2)}$ square invertible $n_2 \times n_2$ matrices and $n = n_1 + n_2$, the blocks of the inverse matrix $\mathbf{M}^{-1}$ can be found by solving the linear system of equations:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{M}^{(1,1)} & \mathbf{M}^{(1,2)} \\ \mathbf{M}^{(2,1)} & \mathbf{M}^{(2,2)} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1 \times n_2} \\ \mathbf{0}_{n_2 \times n_1} & \mathbf{I}_{n_2} \end{bmatrix}.$$

Explicitly solving this system of equations can be pulled apart in solving two independent linear systems of equations. The first one is

$$\begin{cases} \mathbf{A}\mathbf{M}^{(1,2)} + \mathbf{B}\mathbf{M}^{(2,2)} = \mathbf{0}_{n_1 \times n_2} \\ \mathbf{C}\mathbf{M}^{(1,2)} + \mathbf{D}\mathbf{M}^{(2,2)} = \mathbf{I}_{n_2} \end{cases}, \tag{A.1}$$

and because $\mathbf{A}$ is invertible, this translates to

$$\begin{cases} \mathbf{M}^{(1,2)} = -\mathbf{A}^{-1}\mathbf{B}\mathbf{M}^{(2,2)} \\ \left(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)\mathbf{M}^{(2,2)} = \mathbf{I}_{n_2} \end{cases}.$$

The final equations for $\mathbf{M}^{(1,2)}$ and $\mathbf{M}^{(2,2)}$ are thus, assuming that $\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$

is invertible,

$$\begin{cases} \mathbf{M}^{(1,2)} = -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \\ \mathbf{M}^{(2,2)} = \left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1} \end{cases},$$

where $\mathbf{M}^{(2,2)}$ is also the inverse of what is called the Schur complement of $\mathbf{A}$ in $\mathbf{M}$.

The second system can be solved similarly:

$$\begin{cases} \mathbf{A}\mathbf{M}^{(1,1)} + \mathbf{B}\mathbf{M}^{(2,1)} = \mathbf{I}_{n_1} \\ \mathbf{C}\mathbf{M}^{(1,1)} + \mathbf{D}\mathbf{M}^{(2,1)} = \mathbf{0}_{n_2\times n_1} \end{cases} \tag{A.2}$$

$$\Leftrightarrow \begin{cases} \mathbf{M}^{(1,1)} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\mathbf{M}^{(2,1)} \\ \left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)\mathbf{M}^{(2,1)} = -\mathbf{C}\mathbf{A}^{-1} \end{cases}, \tag{A.3}$$

and so the equations for $\mathbf{M}^{(1,1)}$ and $\mathbf{M}^{(2,1)}$ are thus

$$\begin{cases} \mathbf{M}^{(1,1)} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} \\ \mathbf{M}^{(2,1)} = -\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} \end{cases}.$$

To come to these equations we have only made use of the fact that $\mathbf{A}$ is invertible, but when $\mathbf{D}$ is also invertible, these systems of equations can also be solved in another way. The system in Eq. (A.1) can be solved as

$$\begin{cases} \left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)\mathbf{M}^{(1,2)} = -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{M}^{(2,2)} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{C}\mathbf{M}^{(2,1)} \end{cases},$$

leading to, assuming that $\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}$ is invertible,

$$\begin{cases} \mathbf{M}^{(1,2)} = -\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{M}^{(2,2)} = \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1} \end{cases}.$$

The system in Eq. (A.2) is then solved as

$$\begin{cases} \left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)\mathbf{M}^{(1,1)} = \mathbf{I}_{n_1} \\ \mathbf{M}^{(2,1)} = -\mathbf{D}^{-1}\mathbf{C}\mathbf{M}^{(1,1)} \end{cases} \Leftrightarrow \begin{cases} \mathbf{M}^{(1,1)} = \left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} \\ \mathbf{M}^{(2,1)} = -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} \end{cases},$$

where again $\mathbf{M}^{(1,1)}$ is the inverse of the Schur complement of $\mathbf{D}$ in $\mathbf{M}$

The previously derived results can be summarised as follows, leading to two equivalent expressions for the inverse of a block matrix, when both diagonal blocks are invertible:

$$\begin{aligned}\begin{bmatrix}\mathbf{A} & \mathbf{B}\\ \mathbf{C} & \mathbf{D}\end{bmatrix}^{-1} &= \begin{bmatrix}\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\\ -\left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{C}\mathbf{A}^{-1} & \left(\mathbf{D}-\mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\end{bmatrix}\\ &= \begin{bmatrix}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & -\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1}\\ -\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1} & \mathbf{D}^{-1}+\mathbf{D}^{-1}\mathbf{C}\left(\mathbf{A}-\mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right)^{-1}\mathbf{B}\mathbf{D}^{-1}\end{bmatrix},\end{aligned}$$

A.2. Determinant

Some important basic properties of the determinant of a matrix are:

- $|\mathbf{I}_n| = 1$.
- $|\mathbf{A}'| = |\mathbf{A}|$.
- $|\mathbf{AB}| = |\mathbf{A}|\,|\mathbf{B}|$, for square matrices $\mathbf{A}$ and $\mathbf{B}$ of equal dimension.
- $\left|\mathbf{A}^{-1}\right| = |\mathbf{A}|^{-1}$.
- $|k\mathbf{A}| = k^n\,|\mathbf{A}|$ for any scalar k and square matrix $\mathbf{A}$ of dimension n.

The Laplace formula for a determinant of a matrix consists of the sum of its minors [60]. A minor $M_{(i,j)}$ of an $n \times n$ matrix $\mathbf{A}$ is defined as the determinant of the $(n-1)\times(n-1)$ matrix resulting from deleting the ith row and jth column of $\mathbf{A}$:

$$|\mathbf{A}| = \sum_{i=1}^{n}(-1)^{i+j}a_{i,j}M_{i,j} = \sum_{j=1}^{n}(-1)^{i+j}a_{i,j}M_{i,j}\,. \tag{A.4}$$

An important result of this formula is that the determinant of the following matrix

$$\begin{bmatrix}1 & \mathbf{0}_n'\\ \mathbf{x} & \mathbf{A}\end{bmatrix},$$

with $\mathbf{x}$ any vector of dimension n, is equal to the determinant of the square $n \times n$ matrix $\mathbf{A}$. By induction, this can be extended to any matrix of the

type

$$\begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m\times n} \\ \mathbf{B} & \mathbf{A} \end{bmatrix},$$

with $\mathbf{B}$ any matrix of dimension $n \times m$. It was already proven for $m = 1$, so if we assume that it was proven for $m - 1$, then we see that:

$$\det\left(\begin{bmatrix} \mathbf{I}_m & \mathbf{0}_{m\times n} \\ \mathbf{B} & \mathbf{A} \end{bmatrix}\right) = \det\left(\begin{bmatrix} \mathbf{I}_{m-1} & \mathbf{0}_{(m-1)\times n} \\ \mathbf{C} & \mathbf{A} \end{bmatrix}\right) = |\mathbf{A}|\ ,$$

with $\mathbf{C}$ the matrix obtained by deleting the first column of $\mathbf{B}$.

Because $|\mathbf{M}'| = |\mathbf{M}|$, the previous result is also valid for any matrix of the type:

$$\det\left(\begin{bmatrix} \mathbf{I}_m & \mathbf{B}' \\ \mathbf{0}_{m\times n} & \mathbf{A} \end{bmatrix}\right) = |\mathbf{A}|\ .$$

In a similar way, it can be proven that

$$\det\left(\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0}_{m\times n} & \mathbf{I}_m \end{bmatrix}\right) = \det\left(\begin{bmatrix} \mathbf{A} & \mathbf{0}_{n\times m} \\ \mathbf{B}' & \mathbf{I}_m \end{bmatrix}\right) = |\mathbf{A}|\ .$$

The determinant of an arbitrary matrix that is split up in blocks can then be found as

$$|\mathbf{M}| = \det\left(\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}\right) = |\mathbf{A}|\left|\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}\right|\ , \tag{A.5}$$

when $\mathbf{A}$ is invertible by acknowledging that $\mathbf{M}$ can be decomposed as:

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{A} & \mathbf{0}_{n\times m} \\ \mathbf{C} & \mathbf{I}_m \end{bmatrix} \begin{bmatrix} \mathbf{I}_n & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0}_{m\times n} & \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B} \end{bmatrix}.$$

When $\mathbf{D}$ is invertible, it can be analogously derived that

$$|\mathbf{M}| = \det\left(\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}\right) = |\mathbf{D}|\left|\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}\right|\ , \tag{A.6}$$

Another important theorem for the determinant is Sylvester's determinant theorem, stating that for an $n \times m$ matrix $\mathbf{A}$ and an $m \times n$ matrix $\mathbf{B}$:

$$|\mathbf{I}_n + \mathbf{AB}| = |\mathbf{I}_m + \mathbf{BA}| \ .$$

this can immediately be proven by considering the determinant of the block matrix

$$\begin{bmatrix} \mathbf{I}_n & \mathbf{A} \\ \mathbf{B} & \mathbf{I}_m \end{bmatrix} .$$

This determinant can be found by applying Eq. (A.5) as well as Eq. (A.6) and so

$$|\mathbf{I}_n|\,|\mathbf{I}_m - \mathbf{BA}| = |\mathbf{I}_m|\,|\mathbf{I}_n - \mathbf{AB}| \ ,$$

proving Sylvester's determinant theorem.

A.3. Trace

The trace of a square matrix $\mathbf{A}$ of dimension n is defined as the sum of its diagonal values, or more formally

$$\mathrm{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{i,i} \, .$$

This leads to the following properties which are easily verified

- $\mathrm{tr}(\mathbf{A}') = \mathrm{tr}(\mathbf{A})$.
- $\mathrm{tr}(\mathbf{A} + \mathbf{B}) = \mathrm{tr}(\mathbf{A}) + \mathrm{tr}(\mathbf{B})$.
- $\mathrm{tr}(c\mathbf{A}) = c\mathrm{tr}(\mathbf{A})$

The trace of a product of an $n \times m$ matrix $\mathbf{A}$ and an $m \times n$ matrix $\mathbf{B}$ can be expressed as:

$$\mathrm{tr}(\mathbf{AB}) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{i,j}b_{j,i} = \sum_{j=1}^{n}\sum_{i=1}^{n} b_{j,i}a_{i,j} = \mathrm{tr}(\mathbf{BA}) \, .$$

This cyclic property of a trace is an extension on this result, meaning that the trace is invariant under cyclic permutations:

$$\mathrm{tr}(\mathbf{ABCD}) = \mathrm{tr}(\mathbf{BCDA}) = \mathrm{tr}(\mathbf{CDAB}) = \mathrm{tr}(\mathbf{DABC})\,.$$

The trace can also be expressed in function of the eigenvalues of the matrix by making use of the Jordan normal form. The Jordan normal form of a square matrix is

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{J}_2 & \cdots & \mathbf{0} \\ \vdots & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{J}_k \end{bmatrix},$$

where the $\mathbf{J}_i$ are called Jordan blocks that are square matrices of the form

$$\begin{bmatrix} \lambda_i & 1 & \cdots & 0 \\ 0 & \lambda_i & \ddots & \vdots \\ \vdots & & \ddots & 1 \\ 0 & 0 & \cdots & \lambda_i \end{bmatrix},$$

with the dimension d_i of the block the algebraic multiplicity of the eigenvalue λ_i It can be shown that for each square matrix $\mathbf{A}$ there exists an invertible matrix $\mathbf{P}$ such that $\mathbf{P}^{-1}\mathbf{AP} = \mathbf{J}$ [85]. Taking the trace of this expression:

$$\mathrm{tr}(\mathbf{P}^{-1}\mathbf{AP}) = \mathrm{tr}(\mathbf{PP}^{-1}\mathbf{A}) = \mathrm{tr}(\mathbf{A}) = \mathrm{tr}(\mathbf{J}) = \sum_{i=1}^{k} d_i\lambda_i$$

A.4. Derivation rules

For a parameter t, the derivative of a matrix $\mathbf{A}$ with respect to t is the matrix $\mathbf{B}$ with as elements:

$$b_{i,j} = \frac{\partial a_{i,j}}{\partial t} .$$

For an invertible $n \times n$ matrix $\mathbf{A}$, we know that $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}_n$ and taking the derivative of this expression with respect to t leads to

$$\frac{\partial \mathbf{A}}{\partial t}\mathbf{A}^{-1} + \mathbf{A}\frac{\partial \mathbf{A}^{-1}}{\partial t} = \mathbf{0}_{n\times n} ,$$

and so

$$\frac{\partial \mathbf{A}^{-1}}{\partial t} = -\mathbf{A}^{-1}\frac{\partial \mathbf{A}}{\partial t}\mathbf{A}^{-1} .$$

The derivative of the determinant requires the definition of the adjugate matrix $\text{adj}(\mathbf{A})$ of a square matrix $\mathbf{A}$ of dimension n. This adjugate matrix has as elements at position (i, j):

$$\text{adj}(\mathbf{A})_{i,j} = (-1)^{i+j} M_{j,i} ,$$

with $M_{j,i}$ the (j, i)-th minor of $\mathbf{A}$. Using the expression in Eq. (A.4), the following equality holds:

$$\mathbf{A}\,\text{adj}(\mathbf{A}) = |\mathbf{A}|\,\mathbf{I}_n ,$$

and so when $\mathbf{A}$ is invertible

$$\text{adj}(\mathbf{A}) = |\mathbf{A}|\,\mathbf{A}^{-1} .$$

Another prerequisite is the fact that

$$\frac{\partial\,|\mathbf{A}|}{\partial a_{i,j}} = (-1)^{i+j} M_{i,j} = \text{adj}(\mathbf{A})_{j,i} ,$$

which follows from the expression in Eq. (A.4) and the fact that every minor $M_{i,j}$ does not depend on $a_{i,j}$.

The derivative of the determinant of a square matrix $\mathbf{A}$ of dimension n with

respect to a parameter t can then be found by applying the chain rule:

$$\begin{aligned}\frac{\partial |\mathbf{A}|}{\partial t} &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial |\mathbf{A}|}{\partial a_{i,j}} \frac{\partial a_{i,j}}{\partial t} \\ &= \sum_{j=1}^{n}\sum_{i=1}^{n} \operatorname{adj}(\mathbf{A})_{j,i} \frac{\partial a_{i,j}}{\partial t} \\ &= \operatorname{tr}\left(\operatorname{adj}(\mathbf{A}) \frac{\partial \mathbf{A}}{\partial t}\right) .\end{aligned}$$

So when $\mathbf{A}$ is invertible

$$\frac{\partial |\mathbf{A}|}{\partial t} = |\mathbf{A}| \operatorname{tr}\left(\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial t}\right) .$$

As a result the derivative of the logarithm of the determinant of an invertible matrix is

$$\begin{aligned}\frac{\partial \log |\mathbf{A}|}{\partial t} &= \frac{1}{|\mathbf{A}|} \frac{\partial |\mathbf{A}|}{\partial t} \\ &= \operatorname{tr}\left(\mathbf{A}^{-1} \frac{\partial \mathbf{A}}{\partial t}\right) .\end{aligned}$$

A.5. Idempotent matrices

A matrix $\mathbf{T}$ is called idempotent when the product of the multiplication with itself is again itself: $\mathbf{TT} = \mathbf{T}$. This leads to the fact that $\mathbf{T}$ should be square. When $n \times n$ matrix $\mathbf{T}$ is idempotent it can easily be proven that $\mathbf{I}_n - \mathbf{T}$ is also idempotent:

$$\begin{aligned}(\mathbf{I}_n - \mathbf{T})(\mathbf{I}_n - \mathbf{T}) &= \mathbf{I}_n - \mathbf{T} - \mathbf{T} + \mathbf{TT} \\ &= \mathbf{I}_n - 2\mathbf{T} + \mathbf{T} \\ &= \mathbf{I}_n - \mathbf{T} .\end{aligned}$$

Another important property of idempotent matrices is the fact that their eigenvalues are 0 or 1. This follows from

$$\mathbf{Mv} = \mathbf{MMv} = \lambda \mathbf{Mv} = \lambda^2 \mathbf{v} = \lambda \mathbf{v} ,$$

with λ and $\mathbf{v}$ representing each eigenvalue and eigenvector. As such, the only possible values for λ are 0 and 1.

Because it was shown previously that the trace of the matrix is equal to the sum of the eigenvalues, the fact that the eigenvalues can only be 0 or 1 leads to the fact that the trace of an idempotent matrix equals its rank:

$$\operatorname{tr}(\mathbf{T}) = \operatorname{rank}(\mathbf{T})\,.$$

A.6. Quadratic forms

A quadratic form is of the form $\mathbf{y}'\mathbf{A}\mathbf{y}$, with $\mathbf{A}$ generally assumed to be symmetric. If $\mathbf{y}$ is a random vector with expected value $\boldsymbol{\mu}$ and $\operatorname{var}(\mathbf{y}) = \mathbf{V}$ then the expected value of $\mathbf{y}'\mathbf{A}\mathbf{y}$ can be found as:

$$\begin{aligned}
\operatorname{E}(\mathbf{y}'\mathbf{A}\mathbf{y}) &= \operatorname{E}\left(\operatorname{tr}\left(\mathbf{y}'\mathbf{A}\mathbf{y}\right)\right) = \operatorname{E}\left(\operatorname{tr}\left(\mathbf{A}\mathbf{y}\mathbf{y}'\right)\right) \\
&= \operatorname{tr}\left(\operatorname{E}\left(\mathbf{A}\mathbf{y}\mathbf{y}'\right)\right) = \operatorname{tr}\left(\mathbf{A}\operatorname{E}\left(\mathbf{y}\mathbf{y}'\right)\right) \\
&= \operatorname{tr}\left(\mathbf{A}\left(\mathbf{V} + \boldsymbol{\mu}\boldsymbol{\mu}'\right)\right) = \operatorname{tr}\left(\mathbf{A}\mathbf{V}\right) + \operatorname{tr}\left(\mathbf{A}\boldsymbol{\mu}\boldsymbol{\mu}'\right) \\
&= \operatorname{tr}\left(\mathbf{A}\mathbf{V}\right) + \operatorname{tr}\left(\boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}\right) = \operatorname{tr}\left(\mathbf{A}\mathbf{V}\right) + \boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}\,,
\end{aligned}$$

where we have made use of the fact that the trace of a scalar is equal to that scalar, $\operatorname{tr}(\operatorname{E}()) = \operatorname{E}(\operatorname{tr}())$ and $\mathbf{V} = \operatorname{E}(\mathbf{y}\mathbf{y}') - \boldsymbol{\mu}\boldsymbol{\mu}'$.

Dutch summary

Het optimaliseren van de agronomische performantie van planten en dieren is een proces dat sinds vele eeuwen gaande is en waar nog steeds veel aandacht aan wordt besteed. De voorbije eeuwen gebeurde deze optimalisatie door het kruisen van die individuen wiens performantie significant beter was dan de andere individuen, wat ook wel benoemd wordt als fenotypische selectie. Op het einde van de 19de eeuw publiceerde Gregor Mendel de welbekende resultaten van zijn experimenten met doperwten, waaruit hij concludeerde dat bepaalde eigenschappen kunnen doorgegeven worden naar volgende generaties, afhankelijk van onzichtbare factoren die later genen werden genoemd. Het duurde echter tot het midden van de 20ste eeuw totdat deze kennis ook werd toegepast in de plantenveredeling en veefokkerij. Aanvankelijk werd de genetica van de planten en dieren niet expliciet geëxploiteerd, maar werden correlaties verondersteld tussen de fenotypische scores van planten en dieren, gebaseerd op hun onderlinge verwantschap die afgeleid werd van een nauwkeurig bijgehouden stamboom.

Het begin van het huidige millennium bracht echter nieuwe opportuniteiten door enkele belangrijk evoluties in de moleculaire wetenschap van de genetica. De ontdekking van de DNA molecule die de drager is van de genetisch overerfbare informatie en de studie van zijn chemische opbouw leidde het tijdperk in van DNA sequencing, die toelaat om bepaalde sequenties van de DNA molecule te lezen en te ontleden. Al snel werd deze informatie ook aangewend in de plantenveredeling en de veefokkerij, waarbij getracht werd om het effect te schatten van bepaalde genetische merkers, i.e. stukjes DNA sequentie die een grote variabiliteit vertonen in bepaalde populaties, op een fenotypische score. Initieel werden slechts een beperkt aantal van deze merkers opgenomen in de analyse, waarbij men vooral als doel had de plaatsen op het genoom, ook loci genaamd, te ontdekken die een significant effect hebben op een bepaalde fenotypische eigenschap. Deze loci worden dan bestempeld als Quantitative Trait Loci (QTL) en tegenwoordig zijn al heel wat van deze loci bekend. De huidige aanname is echter dat kwantitatieve eigenschappen zoals bijvoorbeeld de opbrengst van maïs niet enkel bepaald wordt door deze QTL met een significant effect, maar dat het gecombineerd effect van genetische merkers met een veel kleinere bijdrage aan de eigenschap

een belangrijke rol kan spelen in de verdere optimalisatie van de performantie van planten en dieren.

Sinds 2008 zijn daarom al commerciële DNA chips beschikbaar die tot 50,000 genetische merkers kunnen samplen over het gehele genoom. Deze genetische informatie wordt gebruikt in de plantenveredeling en veefokkerij om beslissingen te maken over welke individuen er worden gekruist om een volgende, meer performante, generatie te bekomen. Dit selectieproces wordt bestempeld als genoomwijde selectie en het schatten van de genetische component van de eigenschap onder studie wordt genoomwijde voorspelling genoemd. Het spreekt voor zich dat de inclusie van dit grote aantal genetische merkers in de analyse niet enkel leidt tot computationele moeilijkheden, maar ook dat het intrinsiek moeilijker wordt om al deze effecten correct te gaan schatten op basis van een beperkt aantal data punten. Daarom kan een data gedreven aanpak een beter resultaat opleveren in de zoektocht naar een betere schatting van de verschillende effecten die een rol spelen bij het determineren van een kwantitatieve eigenschap. Een andere evolutie die in het voordeel spreekt van een data gedreven aanpak is het feit dat de kost voor het genotyperen van een individu steeds kleiner wordt door de voortdurende technologische vooruitgang in dit gebied.

De beschikbaarheid van zulke grootschalige data sets, waar niet enkel het aantal data punten hoog oploopt, maar waar ook de informatie per data punt aanzienlijk is, leidt tot computationele problemen bij de analyse van deze data sets. Er is dus nood aan het mogelijk maken van zulke grootschalige analyses om de plantenveredeling en veefokkerij naar een volgend niveau te brengen. Deze dissertatie is er dan ook op gericht om de technologische vooruitgang uit de computerwetenschappen toe te passen op het onderzoeksgebied van de genoomwijde voorspelling. De toegang tot krachtige computerclusters is namelijk tegenwoordig niet enkel weggelegd voor de academische wereld, maar ook de industriële sector en zelfs particulieren kunnen gebruik maken van deze zogenaamde supercomputers. Het efficiënt aanwenden van deze rekenkracht kan een grote invloed hebben op de accuraatheid van schattingen van genetische effecten op bepaalde eigenschappen van planten en dieren en kan leiden tot het kweken van gewassen of veesoorten die een optimale performantie hebben onder bepaalde klimatologische omstandigheden.

Deel I van deze dissertatie is gericht op het schetsen van de historische en theoretische context van de gevoerde studie. Er wordt eerst een diepgaand

historisch overzicht gegegeven van de evolutie van genoomwijde voorspelling en selectie, waarna er dieper wordt ingegaan op enkele belangrijk statistische bouwstenen die typisch worden gebruikt in het veld van de genoomwijde selectie. Een eerste belangrijke bouwsteen is het lineaire gemengde model dat gebruik maakt van effecten waarvan verondersteld wordt dat ze getrokken worden uit een probabilistische distributie, in deze dissertatie meer bepaald de normale distributie. Ook wordt aandacht besteed aan het schatten van de variantie van deze probabilistische distributie op basis van de input data. Telkens wordt er ook de nadruk gelegd op de computationele aspecten van het gebruik van deze statistische modellen om een idee te geven wat de oorsprong is van de computationele moeilijkheid van genoomwijde voorspelling. Het eerste deel wordt afgesloten met een overzicht van de aangewende computationele methoden om genoomwijde predictie efficiënt te laten uitvoeren op een hoogperformante rekencluster.

In Deel II worden de geïntroduceerde elementen uit Deel I toegepast voor een applicatie in de veefokkerij en in de plantenveredeling. Eerst worden gedistribueerde rekenmethoden aangewend om een genetische analyse in de veefokkerij mogelijk te maken wanneer voor een groot aantal dieren de genetische informatie beschikbaar is en wanneer het aantal genetische merkers voor het bekomen van deze genetische informatie drastisch stijgt. De implementatie van deze methode wordt dan afgetoetst aan de hand van gesimuleerde data, waaruit blijkt dat het aantal gegenotypeerde individuen een grote invloed heeft op de accuraatheid van de voorspellingen van de kweekwaarden van de dieren. Voor de toepassing in de plantenveredeling worden ook omgevingsfactoren in rekening gebracht en wordt rekening gehouden met het feit dat bepaalde genotypes een verschillend effect kunnen hebben onder variërende omgevingsomstandigheden. Daarvoor worden de methodes aangewend voor de veefokkerij uitgebreid met ijle matrix algebra, wat het mogelijk maakt om grote data sets op een gecomprimeerde manier te verwerken. Opnieuw wordt de implementatie van deze methoden afgetoetst met gesimuleerde data, waarbij de resultaten erop wijzen dat een groter aantal data punten komende uit een bepaalde omgeving leidt tot een accuratere voorspelling van de genetische effecten in deze omgeving.

De conclusies van deze resultaten uit de veefokkerij en de plantenveredeling worden samengevat in Deel III, waar ook enkele toekomstige onderzoeksrichtingen worden aangereikt die dit domein nog verder kunnen helpen in de zoektocht naar de genetische oorsprong van agronomische performantie. Een

veelbelovend onderwerp, waar deze dissertatie een bijdrage toe kan leveren is het optimaliseren van genetische lijnen voor bepaalde weersomstandigheden, waarbij het risico op opbrengstverlies door een verkeerde voorspelling van het weer moet geminimaliseerd worden.

Bibliography

[1] J. Calvert, "Systems biology, synthetic biology and data-driven research: A commentary on Krohs, Callebaut, and O'Malley and Soyer," *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences*, vol. 43, no. 1, pp. 81–84, 2012.

[2] F. M. Clark, "Early ideas on inbreeding and crossbreeding," *The Canadian Veterinary Journal*, vol. 2, no. 9, pp. 329–331, 1961.

[3] G. H. Shull, "A pure-line method in corn breeding," *Annual Report American Breeders Association*, vol. 5, pp. 51–59, 1909.

[4] J. F. Crow, "90 years ago: The beginning of hybrid maize," *Genetics*, vol. 148, no. 3, pp. 923–928, 1998.

[5] R. Bernardo, *Breeding for Quantitative Traits in Plants*. Stemma Press, 2010.

[6] R. A. Fisher, "The correlation between relatives on the supposition of mendelian inheritance," *Transactions of the Royal Society of Edinburgh*, vol. 52, no. 02, pp. 399–433, 1919.

[7] C. R. Henderson, "Estimation of variance and covariance components," *Biometrics*, vol. 9, no. 2, pp. pp. 226–252, 1953.

[8] C. R. Henderson, "Selection index and expected genetic advance," *Statistical Genetics and Plant Breeding*, vol. 982, pp. 141–163, 1963.

[9] C. R. Henderson, "Best linear unbiased estimation and prediction under a selection model," *Biometrics*, vol. 31, no. 2, pp. pp. 423–447, 1975.

[10] H. Geldermann, "Investigations on inheritance of quantitative characters in animals by gene markers I. Methods," *Theoretical and Applied Genetics*, vol. 46, no. 7, pp. 319–330, 1975.

[11] R. L. Fernando and M. Grossman, "Marker assisted selection using best linear unbiased prediction," *Genetics Selection Evolution*, vol. 21, no. 4, p. 467, 1989.

[12] R. Lande and R. Thompson, "Efficiency of marker-assisted selection in the improvement of quantitative traits," *Genetics*, vol. 124, no. 3, pp. 743–756, 1990.

[13] T. H. E. Meuwissen and M. E. Goddard, "The use of marker haplotypes in animal breeding schemes," *Genetics Selection Evolution*, vol. 28, no. 2, p. 161, 1996.

[14] T. H. E. Meuwissen, B. J. Hayes, and M. E. Goddard, "Prediction of total genetic value using genome-wide dense marker maps," *Genetics*, vol. 157, no. 4, pp. 1819–1829, 2001.

[15] L. K. Matukumalli, C. T. Lawley, R. D. Schnabel, J. F. Taylor, M. F. Allan, M. P. Heaton, J. O'Connell, S. S. Moore, T. P. L. Smith, T. S. Sonstegard, and C. P. Van Tassell, "Development and characterization of a high density SNP genotyping assay for cattle," *PLoS ONE*, vol. 4, no. 4, p. e5350, 2009.

[16] P. M. VanRaden, "Efficient methods to compute genomic predictions," *Journal of Dairy Science*, vol. 91, no. 11, pp. 4414–4423, 2008.

[17] A. Nejati-Javaremi, C. Smith, and J. Gibson, "Effect of total allelic relationship on accuracy of evaluation and response to selection," *Journal of Animal Science*, vol. 75, no. 7, pp. 1738–1745, 1997.

[18] M. P. L. Calus and R. F. Veerkamp, "Accuracy of multi-trait genomic selection using different methods," *Genetics Selection Evolution*, vol. 43, no. 1, pp. 1–14, 2011.

[19] J. Nadaf, V. Riggio, T.-P. Yu, and R. Pong-Wong, "Effect of the prior distribution of SNP effects on the estimation of total breeding value," in *BMC proceedings*, vol. 6, p. S6, BioMed Central Ltd, 2012.

[20] G. de los Campos, P. Pérez, A. I. Vazquez, and J. Crossa, "Genome-enabled prediction using the BLR (Bayesian Linear Regression) R-package," in *Genome-Wide Association Studies and Genomic Prediction*, pp. 299–320, Springer, 2013.

[21] B. J. Hayes, P. J. Bowman, A. J. Chamberlain, and M. E. Goddard, "Invited review: Genomic selection in dairy cattle: Progress and challenges," *Journal of Dairy Science*, vol. 92, no. 2, pp. 433–443, 2009.

[22] A. Legarra, C. Robert-Granié, P. Croiseau, F. Guillaume, and S. Fritz, "Improved Lasso for genomic selection," *Genetics Research*, vol. 93, no. 01, pp. 77–87, 2011.

[23] H. D. Daetwyler, M. P. L. Calus, R. Pong-Wong, G. de los Campos, and J. M. Hickey, "Genomic prediction in animals and plants: Simulation of data, validation, reporting, and benchmarking," *Genetics*, vol. 193, no. 2, pp. 347–365, 2013.

[24] S. Maenhout, B. De Baets, and G. Haesaert, "Prediction of maize single-cross hybrid performance: support vector machine regression versus best linear prediction," *Theoretical and Applied Genetics*, vol. 120, no. 2, pp. 415–427, 2010.

[25] R. Bernardo, "Prediction of maize single-cross performance using RFLPs and information from related hybrids," *Crop Science*, vol. 34, no. 1, pp. 20–25, 1994.

[26] D. M. Panter and F. L. Allen, "Using best linear unbiased predictions to enhance breeding for yield in soybean: I. choosing parents," *Crop Science*, vol. 35, no. 2, pp. 397–405, 1995.

[27] J.-B. Denis, H.-P. Piepho, and F. A. Van Eeuwijk, "Modelling expectation and variance for genotype by environment data," *Heredity*, vol. 79, no. 2, pp. 162–171, 1997.

[28] H.-P. Piepho, "Ridge regression and extensions for genomewide selection in maize," *Crop Science*, vol. 49, no. 4, pp. 1165–1176, 2009.

[29] J. Crossa, G. de los Campos, P. Pérez, D. Gianola, J. Burgueño, J. L. Araus, D. Makumbi, R. P. Singh, S. Dreisigacker, J. Yan, V. Arief, M. Banziger, and H.-J. Braun, "Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers," *Genetics*, vol. 186, no. 2, pp. 713–724, 2010.

[30] J. Burgueño, G. de los Campos, K. Weigel, and J. Crossa, "Genomic prediction of breeding values when modeling genotype×environment interaction using pedigree and dense molecular markers," *Crop Science*, vol. 52, no. 2, pp. 707–719, 2012.

[31] T. Schulz-Streeck, J. O. Ogutu, A. Gordillo, Z. Karaman, C. Knaak, and H.-P. Piepho, "Genomic selection allowing for marker-by-

environment interaction," *Plant Breeding*, vol. 132, no. 6, pp. 532–538, 2013.

[32] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, pp. 114–117, Apr. 1965.

[33] K. Meyer, "Approximate accuracy of genetic evaluation under an animal model," *Livestock Production Science*, vol. 21, no. 2, pp. 87–100, 1989.

[34] I. Misztal, "Restricted maximum likelihood estimation of variance components in animal model using sparse matrix inversion and a supercomputer," *Journal of Dairy Science*, vol. 73, no. 1, pp. 163–172, 1990.

[35] A. Legarra and I. Misztal, "Technical note: Computing strategies in genome-wide selection," *Journal of Dairy Science*, vol. 91, no. 1, pp. 360–366, 2008.

[36] I. Misztal, A. Legarra, and I. Aguilar, "Computing procedures for genetic evaluation including phenotypic, full pedigree, and genomic information," *Journal of Dairy Science*, vol. 92, no. 9, pp. 4648–4655, 2009.

[37] H. Gao, O. F. Christensen, P. Madsen, U. S. Nielsen, Y. Zhang, M. S. Lund, and G. Su, "Comparison on genomic predictions using three GBLUP methods and two single-step blending methods in the Nordic Holstein population," *Genetics Selection Evolution*, vol. 44, no. 8, pp. 10–1186, 2012.

[38] S. Tsuruta, I. Misztal, I. Aguilar, and T. J. Lawlor, "Multiple-trait genomic evaluation of linear type traits using genomic and phenotypic data in US Holsteins," *Journal of Dairy Science*, vol. 94, no. 8, pp. 4198–4204, 2011.

[39] Y. Masuda, T. Baba, and M. Suzuki, "Application of supernodal sparse factorization and inversion to the estimation of (co) variance components by residual maximum likelihood," *Journal of Animal Breeding and Genetics*, vol. 131, no. 3, pp. 227–236, 2014.

[40] X. Shen, M. Alam, F. Fikse, and L. Rönnegård, "A novel generalized ridge regression method for quantitative genetics," *Genetics*, vol. 193, no. 4, pp. 1255–1268, 2013.

[41] P. M. VanRaden, C. P. Van Tassell, G. R. Wiggans, T. S. Sonstegard, R. D. Schnabel, J. F. Taylor, and F. S. Schenkel, "Invited review: Reliability of genomic predictions for North American Holstein bulls," *Journal of Dairy Science*, vol. 92, no. 1, pp. 16–24, 2009.

[42] J. B. Cole, S. Newman, F. Foertter, I. Aguilar, and M. Coffey, "Breeding and genetics symposium: Really big data: Processing and analysis of very large data sets," *Journal of Animal Science*, vol. 90, no. 3, pp. 723–733, 2012.

[43] S. R. Searle, G. Casella, and C. E. McCulloch, *Variance Components*, vol. 391. John Wiley & Sons, 1992.

[44] C. Henderson, "Use of relationships among sires to increase accuracy of sire evaluation," *Journal of Dairy Science*, vol. 58, no. 11, pp. 1731 – 1738, 1975.

[45] C. R. Henderson, "A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values," *Biometrics*, vol. 32, no. 1, pp. pp. 69–83, 1976.

[46] D. Habier, R. L. Fernando, K. Kizilkaya, and D. J. Garrick, "Extension of the bayesian alphabet for genomic selection," *BMC Bioinformatics*, vol. 12, no. 1, p. 186, 2011.

[47] C. R. Henderson, "Sire evaluation and genetic trends," *Journal of Animal Science*, vol. 1973, no. Symposium, pp. 10–41, 1973.

[48] C. R. Henderson, O. Kempthorne, S. R. Searle, and C. Von Krosigk, "The estimation of environmental and genetic trends from records subject to culling," *Biometrics*, vol. 15, no. 2, pp. 192–218, 1959.

[49] D. Harville, "Extension of the Gauss-Markov theorem to include the estimation of random effects," *The Annals of Statistics*, pp. 384–395, 1976.

[50] M. Lidauer, I. Strandén, E. Mäntysaari, J. Pösö, and A. Kettunen, "Solving large test-day models by iteration on data and preconditioned conjugate gradient," *Journal of Dairy Science*, vol. 82, no. 12, pp. 2788–2796, 1999.

[51] I. Misztal and D. Gianola, "Indirect solution of mixed model equations," *Journal of Dairy Science*, vol. 70, no. 3, pp. 716–723, 1987.

[52] L. Schaeffer and B. Kennedy, "Computing strategies for solving mixed model equations," *Journal of Dairy Science*, vol. 69, no. 2, pp. 575–579, 1986.

[53] I. Aguilar, I. Misztal, A. Legarra, and S. Tsuruta, "Efficient computation of the genomic relationship matrix and other matrices used in single-step evaluation," *Journal of Animal Breeding and Genetics*, vol. 128, no. 6, pp. 422–428, 2011.

[54] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[55] H. P. Piepho, J. O. Ogutu, T. Schulz-Streeck, B. Estaghvirou, A. Gordillo, and F. Technow, "Efficient computation of ridge-regression best linear unbiased prediction in genomic selection in plant breeding," *Crop Science*, vol. 52, no. 3, pp. 1093–1104, 2012.

[56] H. M. Kang, N. A. Zaitlen, C. M. Wade, A. Kirby, D. Heckerman, M. J. Daly, and E. Eskin, "Efficient control of population structure in model organism association mapping," *Genetics*, vol. 178, no. 3, pp. 1709–1723, 2008.

[57] H. O. Hartley and J. N. Rao, "Maximum-likelihood estimation for the mixed analysis of variance model," *Biometrika*, vol. 54, no. 1-2, pp. 93–108, 1967.

[58] R. A. Fisher, "On the mathematical foundations of theoretical statistics," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 222, no. 594-604, pp. 309–368, 1922.

[59] H. D. Patterson and R. Thompson, "Recovery of inter-block information when block sizes are unequal," *Biometrika*, vol. 58, no. 3, pp. 545–554, 1971.

[60] D. A. Harville, *Matrix algebra from a statistician's perspective*, vol. 1. Springer, 1997.

[61] D. A. Harville, "Maximum likelihood approaches to variance component estimation and to related problems," *Journal of the American Statistical Association*, vol. 72, no. 358, pp. 320–338, 1977.

[62] A. P. Verbyla, "A conditional derivation of residual maximum likelihood," *Australian Journal of Statistics*, vol. 32, no. 2, pp. 227–230, 1990.

[63] J. J. Sylvester, "Xxxvii. on the relation between the minor determinants of linearly equivalent quadratic functions," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 1, no. 4, pp. 295–305, 1851.

[64] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B (methodological)*, pp. 1–38, 1977.

[65] A. R. Gilmour, R. Thompson, and B. R. Cullis, "Average information REML: an efficient algorithm for variance parameter estimation in linear mixed models," *Biometrics*, pp. 1440–1450, 1995.

[66] D. Johnson and R. Thompson, "Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techniques and average information," *Journal of Dairy Science*, vol. 78, no. 2, pp. 449–456, 1995.

[67] A. M. Mathai and S. B. Provost, "Quadratic forms in random variables: theory and applications," *Journal of the American Statistical Association*, 1992.

[68] S. Smith and H.-U. Graser, "Estimating variance components in a class of mixed models by restricted maximum likelihood," *Journal of Dairy Science*, vol. 69, no. 4, pp. 1156–1165, 1986.

[69] I. Misztal and M. Perez-Enciso, "Sparse matrix inversion for restricted maximum likelihood estimation of variance components by expectation-maximization," *Journal of Dairy Science*, vol. 76, no. 5, pp. 1479–1483, 1993.

[70] I. Misztal, "Reliable computing in estimation of variance components," *Journal of Animal Breeding and Genetics*, vol. 125, no. 6, pp. 363–370, 2008.

[71] R. Bernardo and J. Yu, "Prospects for genomewide selection for quantitative traits in maize," *Crop Science*, vol. 47, no. 3, pp. 1082–1090, 2007.

[72] K. Meyer, "PX×AI: Algorithmics for better convergence in restricted maximum likelihood estimation," in *8th World Congress on Genetics Applied to Livestock Production*, 2006.

[73] K. Matilainen, E. A. Mäntysaari, M. H. Lidauer, I. Strandén, and R. Thompson, "Employing a Monte Carlo algorithm in Newton-type methods for restricted maximum likelihood estimation of genetic parameters," *PLoS ONE*, vol. 8, no. 12, p. e80821, 2013.

[74] J. W. Rudan, *The History of Computing at Cornell*, vol. 1. The Internet-First University Press, 2004.

[75] J. Kilby, "Miniaturized electronic circuits," June 23 1964. US Patent 3,138,743.

[76] G. E. Moore, "Progress in digital integrated electronics," *IEDM Tech. Digest*, vol. 11, 1975.

[77] J. E. Thornton, "Considerations in Computer Design Leading up to the Control Data 6600," *Control Data Corp*, 1963.

[78] E. Strohmaier, J. Dongarra, H. Simon, and M. Meuer, "TOP500 Supercomputing sites." `http://www.top500.org/`. Accessed: 2015-09-30.

[79] Mersenne Research, Inc., "Great Internet Mersenne Prime Search." `http://www.mersenne.org/`. Accessed: 2015-09-30.

[80] G. M. Amdahl, "Validity of the single processor approach to achieving large scale computing capabilities," in *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, pp. 483–485, ACM, 1967.

[81] J. L. Gustafson, "Reevaluating Amdahl's law," *Communications of the ACM*, vol. 31, no. 5, pp. 532–533, 1988.

[82] M. Ganal, A. Polley, E.-M. Graner, J. Plieske, R. Wieseke, H. Luerssen, and G. Durstewitz, "Large SNP arrays for genotyping in crop plants," *Journal of Biosciences*, vol. 37, no. 5, pp. 821–828, 2012.

[83] Message Passing Interface Forum, "MPI: A message-passing interface standard," tech. rep., University of Tennessee, Knoxville, TN, USA, 1994.

[84] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for fortran usage," *ACM Transactions on Mathematical Software (TOMS)*, vol. 5, no. 3, pp. 308–323, 1979.

[85] G. H. Golub and C. F. Van Loan, *Matrix Computations*, vol. 3. JHU Press, 2012.

[86] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammerling, A. McKenney, *et al.*, *LAPACK Users' guide*, vol. 9. Siam, 1999.

[87] J. Dongarra, R. Van De Geijn, and D. Walker, "A look at scalable dense linear algebra libraries," in *Scalable High Performance Computing Conference, 1992. SHPCC-92, Proceedings*, pp. 372–379, IEEE, 1992.

[88] B. A. Hendrickson and D. E. Womble, "The torus-wrap mapping for dense matrix calculations on massively parallel computers," *SIAM Journal on Scientific Computing*, vol. 15, no. 5, pp. 1201–1226, 1994.

[89] R. A. Van De Geijn and J. Watts, "SUMMA: Scalable universal matrix multiplication algorithm," *Concurrency-Practice and Experience*, vol. 9, no. 4, pp. 255–274, 1997.

[90] O. Schenk, "PARDISO." `http://www.pardiso-project.org/`. Accessed: 2015-09-30.

[91] The MUMPS team, "MUMPS: a MUltifrontal Massively Parallel sparse direct Solver." `http://mumps.enseeiht.fr/`. Accessed: 2015-09-30.

[92] The Numerical Analysis Group at the STFC Rutherford Appleton Laboratory, "The HSL Mathematical Software Library." `http://www.hsl.rl.ac.uk/`. Accessed: 2015-09-30.

[93] X. S. Li, J. Demmel, J. Gilbert, L. Grigori, P. Sao, M. Shao, and I. Yamazaki, "SuperLU: Supernodal LU." `http://crd-legacy.lbl.gov/~xiaoye/SuperLU/`. Accessed: 2015-09-30.

[94] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Siam, 2003.

[95] J. W. Liu, "The multifrontal method for sparse matrix solution: Theory and practice," *SIAM Review*, vol. 34, no. 1, pp. 82–109, 1992.

[96] E. Cuthill and J. McKee, "Reducing the bandwidth of sparse symmetric matrices," in *Proceedings of the 1969 24th National Conference*, pp. 157–172, ACM, 1969.

[97] H. M. Markowitz, "The elimination form of the inverse and its application to linear programming," *Management Science*, vol. 3, no. 3, pp. 255–269, 1957.

[98] P. R. Amestoy, T. A. Davis, and I. S. Duff, "An approximate minimum degree ordering algorithm," *SIAM Journal on Matrix Analysis and Applications*, vol. 17, no. 4, pp. 886–905, 1996.

[99] A. George, "Nested dissection of a regular finite element mesh," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 345–363, 1973.

[100] G. Karypis and K. Schloegel, "ParMETIS - Parallel Graph Partitioning and Fill-reducing Matrix Ordering." `http://glaros.dtc.umn.edu/gkhome/metis/parmetis/overview/`. Accessed: 2015-09-30.

[101] F. Pellegrini, "PT-Scotch: Software package and libraries for parallel graph partitioning, static mapping and clustering, sequential mesh and hypergraph partitioning, and sequential and parallel sparse matrix block ordering." `https://www.labri.fr/perso/pelegrin/scotch/`. Accessed: 2015-09-30.

[102] B. M. Irons, "A frontal solution program for finite element analysis," *International Journal for Numerical Methods in Engineering*, vol. 2, no. 1, pp. 5–32, 1970.

[103] I. S. Duff and J. K. Reid, "The multifrontal solution of indefinite sparse symmetric linear," *ACM Transactions on Mathematical Software (TOMS)*, vol. 9, no. 3, pp. 302–325, 1983.

[104] K. Takahashi, "Formation of sparse bus impedance matrix and its application to short circuit study," in *Proc. PICA Conference, June, 1973*, 1973.

[105] A. Erisman and W. Tinney, "On computing certain elements of the inverse of a sparse matrix," *Communications of the ACM*, vol. 18, no. 3, pp. 177–179, 1975.

[106] Y. E. Campbell and T. A. Davis, "Computing the sparse inverse subset: an inverse multifrontal approach," *University of Florida, Gainesville, FL, Tech. Rep. TR-95-021*, 1995.

[107] M. Perez-Enciso, I. Misztal, and M. Elzo, "FSPAK: An interface for public domain sparse matrix subroutines," in *Proc. 5th World Congr. Genet. Appl. Livest. Prod*, vol. 22, pp. 87–88, 1994.

[108] I. Misztal and collaborators, "BLUPF90 Family of Programs." `http://nce.ads.uga.edu/wiki`. Accessed: 2015-09-30.

[109] I. Aguilar, I. Misztal, D. Johnson, A. Legarra, S. Tsuruta, and T. Lawlor, "Hot topic: A unified approach to utilize phenotypic, full pedigree, and genomic information for genetic evaluation of holstein final score," *Journal of Dairy Science*, vol. 93, no. 2, pp. 743–752, 2010.

[110] A. Legarra and V. Ducrocq, "Computational strategies for national integration of phenotypic, genomic, and pedigree data in a single-step best linear unbiased prediction," *Journal of Dairy Science*, vol. 95, no. 8, pp. 4629–4645, 2012.

[111] V. Wimmer, T. Albrecht, H.-J. Auinger, and C.-C. Schön, "synbreed: a framework for the analysis of genomic prediction data using R," *Bioinformatics*, vol. 28, no. 15, pp. 2086–2087, 2012.

[112] J. M. Hickey and G. Gorjanc, "Simulated data for genomic selection and genome-wide association studies using a combination of coalescent and gene drop methods," *G3: Genes|Genomes|Genetics*, vol. 2, no. 4, pp. 425–427, 2012.

[113] D.-J. de Koning and L. McIntyre, "Setting the standard: a special focus on genomic selection in GENETICS and G3," *Genetics*, vol. 190, no. 4, pp. 1151–1152, 2012.

[114] G. K. Chen, P. Marjoram, and J. D. Wall, "Fast and flexible simulation of DNA sequence data," *Genome Research*, vol. 19, no. 1, pp. 136–142, 2009.

[115] K. A. Wetterstrand, "DNA sequencing costs: data from the NHGRI Genome Sequencing Program (GSP)." `http://www.genome.gov/sequencingcosts`. Accessed: 2015-10-20.

[116] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI-The Complete Reference, Volume 1: The MPI Core*. Cambridge, MA, USA: MIT Press, 2nd. (revised) ed., 1998.

[117] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. W. Walker, and R. C. Whaley, "A proposal for a set of parallel basic linear algebra subprograms," in *Proceedings of the Second International Workshop on Applied Parallel Computing, Computations in Physics, Chemistry and Engineering Science*, PARA '95, (London, UK, UK), pp. 107–114, Springer-Verlag, 1996.

[118] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley, *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, 1997.

[119] The HDF Group, "Hierarchical data format version 5." `http://www.hdfgroup.org/HDF5`. Accessed: 2014-04-24.

[120] D. Habier, R. L. Fernando, and D. J. Garrick, "Genomic BLUP decoded: a look into the black box of genomic prediction," *Genetics*, vol. 194, no. 3, pp. 597–607, 2013.

[121] S. Tsuruta, I. Misztal, and T. Lawlor, "Short communication: Genomic evaluations of final score for US Holsteins benefit from the inclusion of genotypes on cows," *Journal of Dairy Science*, vol. 96, no. 5, pp. 3332–3335, 2013.

[122] S. König, G. Dietl, I. Raeder, and H. H. Swalve, "Genetic relationships for dairy performance between large-scale and small-scale farm conditions," *Journal of Dairy Science*, vol. 88, pp. 4087–4096, 11 2005.

[123] H. A. Mulder and P. Bijma, "Effects of genotype×environment interaction on genetic gain in breeding programs1," *Journal of Animal Science*, vol. 83, pp. 49–61, 01 2005.

[124] A. De Coninck, J. Fostier, S. Maenhout, and B. De Baets, "DAIRRy-BLUP: A high-performance computing approach to genomic prediction," *Genetics*, vol. 197, no. 3, pp. 813–822, 2014.

[125] M. Cooper, D. W. Podlich, and O. S. Smith, "Gene-to-phenotype models and complex trait genetics," *Crop and Pasture Science*, vol. 56, no. 9, pp. 895–918, 2005.

[126] T. Schulz-Streeck, J. O. Ogutu, and H.-P. Piepho, "Comparisons of single-stage and two-stage approaches to genomic selection," *Theoretical and Applied Genetics*, vol. 126, no. 1, pp. 69–82, 2013.

[127] N. Heslot, D. Akdemir, M. E. Sorrells, and J.-L. Jannink, "Integrating environmental covariates and crop modeling into the genomic selection framework to predict genotype by environment interactions," *Theoretical and Applied Genetics*, vol. 127, no. 2, pp. 463–480, 2014.

[128] M. Lopez-Cruz, J. Crossa, D. Bonnett, S. Dreisigacker, J. Poland, J.-L. Jannink, R. P. Singh, E. Autrique, and G. de los Campos, "Increased prediction accuracy in wheat breeding trials using a marker×environment interaction genomic selection model," *G3: Genes/Genomes/Genetics*, vol. 5, no. 4, pp. 569–582, 2015.

[129] N. Heslot, H.-P. Yang, M. E. Sorrells, and J.-L. Jannink, "Genomic selection in plant breeding: a comparison of models," *Crop Science*, vol. 52, no. 1, pp. 146–160, 2012.

[130] D. Habier, R. L. Fernando, and J. C. M. Dekkers, "The impact of genetic relationship information on genome-assisted breeding values," *Genetics*, vol. 177, no. 4, pp. 2389–2397, 2007.

[131] O. F. Christensen and M. S. Lund, "Genomic prediction when some animals are not genotyped," *Genetics Selection Evolution*, vol. 42, no. 2, pp. 1–8, 2010.

[132] D. Jarquín, J. Crossa, X. Lacaze, P. Du Cheyron, J. Daucourt, J. Lorgeou, F. Piraux, L. Guerreiro, P. Pérez, M. Calus, J. Burgueño, and G. de los Campos, "A reaction norm model for genomic selection using high-dimensional genomic and environmental data," *Theoretical and Applied Genetics*, vol. 127, no. 3, pp. 595–607, 2014.

[133] D. W. Podlich, C. R. Winkler, and M. Cooper, "Mapping as you go: An effective approach for marker-assisted selection of complex traits," *Crop Science*, vol. 44, no. 5, pp. 1560–1571, 2004.

[134] M. Cooper, S. C. Chapman, D. W. Podlich, and G. L. Hammer, "The GP problem: quantifying gene-to-phenotype relationships," *Silico Biology*, vol. 2, pp. 151–164, 2002.

[135] C. C. Schön, H. F. Utz, S. Groh, B. Truberg, S. Openshaw, and A. E. Melchinger, "Quantitative trait locus mapping based on resampling

in a vast maize testcross experiment and its relevance to quantitative genetics for complex traits," *Genetics*, vol. 167, no. 1, pp. 485–498, 2004.

[136] H. P. Piepho, "Statistical tests for QTL and QTL-by-environment effects in segregating populations derived from line crosses," *Theoretical and Applied Genetics*, vol. 110, no. 3, pp. 561–566, 2005.

[137] F. A. van Eeuwijk, M. Malosetti, X. Yin, P. C. Struik, and P. Stam, "Statistical models for genotype by environment data: from conventional ANOVA models to eco-physiological QTL models," *Crop and Pasture Science*, vol. 56, no. 9, pp. 883–894, 2005.

[138] M. P. Boer, D. Wright, L. Feng, D. W. Podlich, L. Luo, M. Cooper, and F. A. van Eeuwijk, "A mixed-model quantitative trait loci (QTL) analysis for multiple-environment trial data using environmental covariables for QTL-by-environment interactions, with an example in maize," *Genetics*, vol. 177, no. 3, pp. 1801–1813, 2007.

[139] J. M. Hickey, G. Gorjanc, S. Hearne, and B. E. Huang, "AlphaMPSim: flexible simulation of multi-parent crosses," *Bioinformatics*, vol. 30, no. 18, pp. 2686–2688, 2014.

[140] J. Crossa, P. Perez, J. Hickey, J. Burgueno, L. Ornella, J. Ceron-Rojas, X. Zhang, S. Dreisigacker, R. Babu, Y. Li, D. Bonnett, and K. Mathews, "Genomic prediction in CIMMYT maize and wheat breeding programs," *Heredity*, vol. 112, no. 1, pp. 48–60, 2014.

[141] C. Shindo, H. Tsujimoto, and T. Sasakuma, "Segregation analysis of heading traits in hexaploid wheat utilizing recombinant inbred lines," *Heredity*, vol. 90, no. 1, pp. 56–63, 2003.

[142] W. T. Federer and D. Raghavarao, "On augmented designs," *Biometrics*, pp. 29–35, 1975.

[143] A. Kuzmin, M. Luisier, and O. Schenk, "Fast methods for computing selected elements of the greens function in massively parallel nanoelectronic device simulations," in *Euro-Par 2013 Parallel Processing* (F. Wolf, B. Mohr, and D. Mey, eds.), vol. 8097 of *Lecture Notes in Computer Science*, pp. 533–544, Springer Berlin Heidelberg, 2013.

[144] O. Schenk, M. Bollhöfer, and R. A. Römer, "On large-scale diagonalization techniques for the anderson model of localization," *SIAM Review*, vol. 50, pp. 91–112, Feb. 2008.

[145] O. Schenk, A. Wächter, and M. Hagemann, "Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization," *Computational Optimization and Applications*, vol. 36, no. 2-3, pp. 321–341, 2007.

[146] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[147] G. de los Campos, J. M. Hickey, R. Pong-Wong, H. D. Daetwyler, and M. P. L. Calus, "Whole-genome regression and prediction methods applied to plant and animal breeding," *Genetics*, vol. 193, no. 2, pp. 327–345, 2013.

[148] F. A. van Eeuwijk, M. Malosetti, and M. P. Boer, "Modelling the genetic basis of response curves underlying genotype×environment interaction," in *Scale and Complexity in Plant Systems Research: Gene-Plant-Crop Relations* (J. H. J. Spiertz, P. C. Struik, and H. H. van Laar, eds.), vol. 21 of *Wageningen UR Frontis Series*, pp. 113–124, Springer, Berlin/Heidelberg, Germany/New York, 2007.

[149] L. Moreau, A. Charcosset, and A. Gallais, "Use of trial clustering to study QTL×environment effects for grain yield and related traits in maize," *Theoretical and Applied Genetics*, vol. 110, no. 1, pp. 92–105, 2004.

[150] T. Schulz-Streeck, J. Ogutu, and H.-P. Piepho, "Pre-selection of markers for genomic selection," *BMC Proceedings*, vol. 5, no. Suppl 3, p. S12, 2011.

[151] Council on Dairy Cattle Breeding, "Genotype counts by Chip Type, Breed Code, and Sex Codes." `https://www.cdcb.us/Genotype/cur_freq.html`. Accessed: 2015-09-30.

[152] R. van Binsbergen, M. P. Calus, M. C. Bink, F. A. van Eeuwijk, C. Schrooten, and R. F. Veerkamp, "Genomic prediction using imputed whole-genome sequence data in holstein friesian cattle," *Genetics Selection Evolution*, vol. 47, no. 1, pp. 1–13, 2015.

[153] A. Benoit, S. K. Raina, and Y. Robert, "Efficient checkpoint/verification patterns," *International Journal of High Performance Computing Applications*, p. 1094342015594531, 2015.

[154] T. Herault and Y. Robert, "Fault-tolerance techniques for high-performance computing," *Computer Communications*, 2015.

[155] J. Dongarra, J. Kurzak, P. Luszczek, and I. Yamazaki, "PULSAR: Parallel Ultra-Light Systolic Array Runtime." `http://icl.cs.utk.edu/pulsar/index.html`. Accessed: 2015-09-30.

[156] M. Köhler and J. Saak, "FlexiBLAS - A flexible BLAS library with runtime exchangeable backends," Tech. Rep. 284, LAPACK Working Note, jan 2014.

[157] A. Buluç and J. R. Gilbert, "The combinatorial blas: Design, implementation, and applications," *International Journal of High Performance Computing Applications*, vol. 25, no. 4, pp. 496–509, 2011.

[158] J. R. Bunch and L. Kaufman, "Some stable methods for calculating inertia and solving symmetric linear systems," *Mathematics of Computation*, pp. 163–179, 1977.

[159] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. Knepley, *et al.*, "Petsc users manual revision 3.5," tech. rep., Argonne National Laboratory (ANL), 2014.

[160] Y. Masuda, I. Misztal, S. Tsuruta, D. A. Lourenco, B. O. Fragomeni, A. Legarra, I. Aguilar, and T. J. Lawlor, "Single-step genomic evaluations with 570k genotyped animals in us holsteins," *Interbull Bulletin*, no. 49, 2015.

Curriculum Vitae

Personalia

Name	Arne De Coninck
Gender	Male
Date of birth	07/11/1985
Place of birth	Zottegem, Belgium
E-mail	Arne.DeConinck@UGent.be

Education

University

- 2007 - 2008

  Third year of second cycle Engineering physics (option applied physics), Ghent University.

- 2006 - 2007

  Second year of second cycle Engineering physics (option applied physics), Ghent University.

- 2005 - 2006

  First year of second cycle Engineering physics, Ghent University.

- 2003 - 2005

  First cycle Engineering sciences, Ghent University.

Secondary school

- 1997 - 2003:

  Koninklijk Lyceum Aalst, Aalst, Belgium

Employment

- 2011 - present

  PhD student at research unit KERMIT, department of Mathematical modelling, statistics and bioinformatics, Ghent University.

- 2009 - 2011

  Graduate Engineer at Core & fuel studies unit, nuclear department, Tractebel Engineering.

Scientific output

Publications in international journals (ISI-papers)

- **De Coninck, A.**, De Baets, B., Kourounis, D., Verbosio, F., Schenk, O., Maenhout, S., and Fostier, J. (2015). Needles: towards large-scale genomic prediction with marker-by-environment interaction. Submitted to: GENETICS.
- **De Coninck, A.**, Fostier, J., Maenhout, S., and De Baets, B. (2014). DAIRRy-BLUP: a high-performance computing approach to genomic prediction. GENETICS, 197(3), 813-822.

Conference proceedings

- **De Coninck, A.**, Verbosio, F., Korounis, D., Schenk, O., Fostier, J., Maenhout, S., and De Baets, B. (2015). Towards parallel large-

scale genomic prediction by coupling sparse and dense matrix algebra. Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP), 747-750. Presented at the 23rd Conference on Parallel, Distributed, and Network-Based Processing (PDP 2015), Turku, Finland.

- **De Coninck, A.**, Verbosio, F., Korounis, D., Schenk, O., De Baets, B., Maenhout, S., and Fostier, J. (2015). Including explicit marker-by-environment interaction for large-scale genomic prediction. Communications in Agricultural and Applied Biological Sciences, 80(1), 117-121. Presented at the 20th National Symposium on Applied Biological Sciences (NSABS 2015), Louvain-La-Neuve, Belgium.

- **De Coninck, A.**, Fostier, J., Maenhout, S., and De Baets, B. (2014). A high-performance computing approach for genomic prediction. Communications in Agricultural and Applied Biological Sciences, 79(1), 115-119. Presented at the 19th National Symposium on Applied Biological Sciences (NSABS 2014), Gembloux, Belgium.

Conference abstracts

- **De Coninck, A.**, Fostier, J., Maenhout, S., and De Baets, B. (2015). A parallel implementation of genomic prediction including genetic by environment interaction. Presented at the 11th ISCB Student Council Symposium. Dublin, Ireland.