University of Colorado, Boulder
CU Scholar

Applied Mathematics Graduate Theses & Dissertations

Applied Mathematics

Fall 11-28-2018

Stochastic Lanczos Likelihood Estimation of Genomic Variance Components

Richard Border
Richard.Border@Colorado.EDU

Follow this and additional works at: https://scholar.colorado.edu/appm_gradetds

 Part of the Biostatistics Commons, and the Numerical Analysis and Computation Commons

Recommended Citation
Border, Richard, "Stochastic Lanczos Likelihood Estimation of Genomic Variance Components" (2018). *Applied Mathematics Graduate Theses & Dissertations*. 120.
https://scholar.colorado.edu/appm_gradetds/120

This Thesis is brought to you for free and open access by Applied Mathematics at CU Scholar. It has been accepted for inclusion in Applied Mathematics Graduate Theses & Dissertations by an authorized administrator of CU Scholar. For more information, please contact cuscholaradmin@colorado.edu.

Stochastic Lanczos Likelihood Estimation of Genomic Variance Components

by

Richard Border

Department of Applied Mathematics

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Applied Mathematics

2018

This thesis entitled:
Stochastic Lanczos Likelihood Estimation of Genomic Variance Components
written by Richard Border
has been approved for the Department of Applied Mathematics

Stephen Becker

Jem Corcoran

Matthew Keller

Date _____

The final copy of this thesis has been examined by the signatories, and we find that
both the content and the form meet acceptable presentation standards of scholarly work
in the above mentioned discipline.

Border, Richard Schuster (M.S., Applied Mathematics)

Stochastic Lanczos Likelihood Estimation of Genomic Variance Components

Thesis directed by Assistant Professor Stephen Becker

Genomic variance components analysis seeks to estimate the extent to which interindividual variation in a given trait can be attributed to genetic similarity. Likelihood estimation of such models involves computationally expensive operations on large, dense, and unstructured matrices of high rank. As a result, standard estimation procedures relying on direct matrix methods become prohibitively expensive as sample sizes increase. We propose a novel estimation procedure that uses the Lanczos process and stochastic Lanczos quadrature to approximate the likelihood for an initial choice of parameter values. Then, by identifying the variance components parameter space with a family of shifted linear systems, we are able to exploit the Krylov subspace shift-invariance property to efficiently compute the likelihood for all additional parameter values of interest in linear time. Numerical experiments using simulated data demonstrate increased performance relative to conventional methods with little loss of accuracy.

Contents

1 Introduction **1**

1.1 Motivation and problem overview 1

1.2 Heritability analysis in complex traits genetics 2

1.3 Defining heritability 2

1.3.1 Twin models 3

1.3.2 Genomic variance components models 4

1.3.3 Applications of heritability analysis 4

1.3.4 What heritability is not 5

1.4 The difficulty of genomic variance components analysis 5

1.5 Previous work and our contributions 8

1.6 Outline of thesis 9

2 Likelihood estimation of variance components **10**

2.1 Introduction 10

2.2 Notation and general model 10

2.3 ML estimation of variance components 11

2.4 Prediction of latent variables 12

2.5 Newton-type algorithms for ML estimation 13

2.6 Looking ahead 14

3 Optimizing performance of estimation algorithms **15**

3.1 Overview 15

3.2 Spectral decomposition based Newton Raphson 15

3.3 Monte Carlo Newton Raphson 18

3.4 Improving existing methods 21

4 Krylov subspace methods **22**

4.1 Overview . . . 22

4.2 The Lanczos process . . . 24

4.3 Block Lanczos Conjugate Gradients (BLCG) . . . 26

4.4 Shifted systems and BLCG . . . 27

5 Approximating the log determinant **31**

5.1 Overview . . . 31

5.2 Stochastic trace estimation . . . 32

5.3 Lanczos polynomials and Lanczos Quadrature . . . 33

5.4 Stochastic Lanczos Quadrature to approximate $\log\det A$ . . . 34

6 Proposed algorithm **37**

6.1 Putting everything together . . . 37

6.2 A Lanczos method for genomics variance components estimation . . . 37

6.3 Performance and numerical experiments . . . 40

7 Conclusion **43**

7.1 Summary . . . 43

7.2 Limitations and future directions . . . 44

Bibliography **46**

Appendix A Notation and glossary **49**

A.1 Symbols . . . 49

A.2 Abbreviations . . . 50

Appendix B Identities in multivariate statistics and matrix algebra **52**

Appendix C Software implementations **54**

C.1 Libraries . . . 54

C.2 Spectral Newton Raphson . . . 54

C.3 Monte Carlo Newton Raphson . . . 55

C.4 Block Lanczos Conjugate Gradient update for seed systems (BLCG-seed) . . . 56

C.5 Block Lanczos Conjugate Gradient update for shifted systems (BLCG-update) . . . 57

C.6 Stochastic Lanczos quadrature (SLQ) to approximate log det from BLCG-seed output 59

C.7 Stochastic Block Lanczos Likelihood Estimation (SBLLE) 59

C.8 Data simulation 62

List of Figures

1.1 Density/sparsity of the genetic covariance structures in genomic variance components versus the classical twin model 7
1.2 Condition number of covariance matrix for varying h^2 8
4.1 Spectrum of covariance matrix and convergence of conjugate gradients for varying h^2 23
4.2 Performance of BLCG-seed and BLCG-update (log scale) 29
4.3 Performance of BLCG-seed and BLCG-update 30
5.1 SLQ log determinant approximation accuracy 36
6.1 Timings of complete SBLLE versus single eigendecomposition (log scale) 41
6.2 Timings of complete SBLLE versus single eigendecomposition 41
6.3 Accuracy of stochastic algorithms with respect to exact algorithms 42

Chapter 1

Introduction

1.1 Motivation and problem overview

The decomposition of phenotypic variance into heritable and non-heritable components has been of primary interest in quantitative genetics since the field's inception [4, 19, 5]. Prior to the previous decade, such analyses were restricted to pedigree-type data collected from sets of related and unrelated individuals (e.g., fraternal and identical twin pairs). In recent years, the increasing affordability of genotyping, as well as the increasing size of genotyped samples, has permitted the extension of such methods to incorporate directly measured genotypes into the examination of individual differences among unrelated individuals [28].

Estimating such models, which we refer to as *genomic variance component models*, typically involves computationally intensive matrix operations on large, dense, high rank matrices with minimal structure which arise in the evaluation of a likelihood or its derivatives. Samples of interest are already large enough (e.g., ≈500,000 [24] to ≈1,100,000 [14] participants) to render direct methods for matrix operations (i.e., $O(n^3)$ methods for solving linear systems or evaluating bilinear forms) infeasible and sample sizes will only continue to grow in the future. As a result, computationally efficient estimation of genomic variance components models had attracted considerable interest in recent years [15, 29, 18, 16], with efforts directed toward minimizing the number of $O(n^3)$ operations, approximate inference, and/or the substitution of iterative matrix methods for direct methods. In the present thesis, we extend previous efforts in the latter two directions by incorporating recent advances in numerical linear algebra and by exploiting structure specific to the problem at hand.

In the following introductory discussion, we consider our subject in the broader contexts of quantitative genetics and general variance components analysis. We then emphasize the unique difficulties encountered in genomic variance component estimation with respect to these canonical

methods. Finally, we provide an outline of the rest of the thesis, wherein we motivate and propose methods for addressing these difficulties.

1.2 Heritability analysis in complex traits genetics

Before discussing the notion of heritability, it is helpful to explain what is meant by the phrase "complex traits." The complexity of said traits does not lie in their definitions or measurement; for example, height, which is among the easiest to define and measure of all human traits, is *genetically complex* in that substantial interindividual variation cannot be explained by variation at a small number of polymorphic loci. Instead, height is thought to be highly *polygenic*; that is, its heritable component reflects variation at many hundreds or thousands of loci, each exerting extremely small, difficult-to-detect effects [28]. In fact, most *common* traits of interest to behavioral and health researchers (i.e., excluding rare, monogenic disorders) appear to be highly polygenic.

1.3 Defining heritability

Heritability (h^2) can be briefly defined as the fraction of variance of a given phenotype attributable to additive genetic effects[1]. In general, we represent a phenotype by a random vector y with marginal distribution of the form

$$y \sim \mathcal{MVN}(\mu, \sigma_k^2 K + \sigma_r^2 R) \tag{1.3.1}$$

Here, the kinship matrix $K \in \mathbb{S}_+^n$[2] is a correlation matrix describing genetic similarity between individuals and $R \in \mathbb{S}_{++}^n$ (which is often simply the identity matrix) similarly reflects non-genetic similarity or dissimilarity between individuals. The parameters $\sigma_k^2, \sigma_r^2 > 0$, $\sigma_k^2 + \sigma_r^2 = 1$, comprise the heritable and non-heritable variance components, respectively (we assume, without loss of generality, that $Var(y) = 1$).

[1] Technically, we might call this quantity "narrow sense heritability" as it does not include potential non-additive genetic effects, including dominance or epistasis

[2] in the genomic case, we may assume $K \in \mathbb{S}_{++}^n$

1.3.1 Twin models

The classical twin model serves as an intuitive example of variance components models as employed in quantitative genetics [4]. Represent a phenotype by the random vector y with marginal distribution

$$y \sim \mathcal{MVN}(\mu, \sigma_a^2 A + \sigma_c^2 C + \sigma_e^2 I), \tag{1.3.2}$$

where

$$A = \begin{pmatrix} 1 & & & & \\ \underset{\text{MZ}}{1} & 1 & & \text{Sym} & \\ & & \ddots & & \\ & \underset{\text{unrel}}{0} & & 1 & \\ & & & \underset{\text{DZ}}{1/2} & 1 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & & & & \\ \underset{\text{rel}}{1} & 1 & & \text{Sym} & \\ & & \ddots & & \\ & \underset{\text{unrel}}{0} & & 1 & \\ & & & \underset{\text{rel}}{1} & 1 \end{pmatrix}. \tag{1.3.3}$$

The matrix A represents the expected covariance between individuals due to additive genetic factors: identical, or monozygotic (MZ) twins are identical in genotype; on the other hand, fraternal, or dizygotic (DZ) twins, are on average identical at only 50% of polymorphic loci; finally, we assume individuals from different families are on average independent at polymorphic loci. These differences determine the expected contributions of additive genetic effects to the similarity between individuals with respect to y under random mating [5].

The matrix C accounts for the nesting of twin pairs within families: similarity among related individuals (rel), in this case siblings, might reflect similarity in rearing environment or other non-genetic factors that would not be as such for unrelated (unrel) individuals.

Finally, the identity matrix accounts for residual variation unique to each individual beyond that attributable to additive genetic effects and common environment. The variance components $\sigma_a^2, \sigma_c^2, \sigma_e^2$, which are to be estimated, reflect the decomposition of the total covariance of y into that attributable to additive genetic variation, shared environmental variation, and residual variation, respectively. To summarize,

$$\begin{aligned} \underset{i \neq j}{Cov}(y_i, y_j) =& \sigma_a^2 \left[\!\left[i, j \text{ form an MZ pair} \right]\!\right] + \frac{\sigma_a^2}{2} \left[\!\left[i, j \text{ form an DZ pair} \right]\!\right] \\ &+ \sigma_c^2 \left[\!\left[i, j \text{ form a sibling pair} \right]\!\right] + \sigma_e^2. \end{aligned} \tag{1.3.4}$$

An estimate of the heritability of y with unit variance derived from a fitted twin model is simply the estimate of additive genetic variance component $\hat{h}^2_{\text{twin}} = \hat{\sigma}^2_a$. In general, when $Var(y) \neq 1$, the heritability is estimated as the fraction of total variance comprised by the additive component: $\hat{h}^2_{\text{twin}} = \hat{\sigma}^2_a/(\hat{\sigma}^2_a + \hat{\sigma}^2_c + \hat{\sigma}^2_e)$.

1.3.2 Genomic variance components models

The genomic variance components model, in contrast to pedigree-based models such as the classical twin model, utilizes measured genotypes to index genetic similarity, in samples of unrelated individuals[3]. Let $Z \in \mathbb{R}^{n \times m}$, $m \gg n$, be a standardized genotype rank n matrix consisting of n individuals' genotypes at m loci, where each column has zero mean and unit variance. The genomic relatedness matrix (GRM) $\frac{1}{m}ZZ^T$ plays a role analogous to that of the A matrix in the classical twin model (1.3.2):

$$y \sim \mathcal{MVN}(\mu, \underbrace{\sigma^2_g}_{\substack{\text{heritable} \\ \text{comp.}}} \underbrace{\frac{1}{m}ZZ^T}_{\text{GRM}} + \underbrace{\sigma^2_e}_{\substack{\text{non-} \\ \text{heritable} \\ \text{comp.}}} I_n). \tag{1.3.5}$$

Note that there is no common environment variance component σ^2_c as we presume the individuals' rearing environments were independent of one another. An estimate of the heritability of y with unit variance derived from a genomic variance components is the estimate of additive genetic variance component $\hat{h}^2_{\text{SNP}} = \hat{\sigma}^2_g$. In general, the heritability is estimated as the fraction of total variance comprised by the additive component: $\hat{h}^2_{\text{SNP}} = \hat{\sigma}^2_g/(\hat{\sigma}^2_g + \hat{\sigma}^2_e)$. The subscript "SNP" reflects the fact that columns of Z are typically comprised of measured genotypes at single nucleotide polymorphism (SNP) loci; further details of the underlying biology aren't necessary for the present thesis.

1.3.3 Applications of heritability analysis

To many researchers, estimating genetic variance components has merit in its own right. Particular topics of interest might include examining changes in heritability over time and/or environment or "partitioned" heritability analyses, wherein the heritable variance component is further decomposed according to constructs of interest, such as risk allele frequency or other biological

[3] i.e., individuals sharing no recent common ancestors

annotations. Alternatively, variance components are often estimated in the context of linear mixed effects models in association studies. In these cases, the variance components and relatedness information are used to account for the confounding effects of population stratification by using genetic relatedness to specify the residual covariance between observations [27]. Though the present work does not focus on any of these directions, the methods we propose are applicable or extensible to such lines of inquiry.

1.3.4 What heritability is not

Though the present thesis is not concerned with the substantive interpretation of variance components with respect to complex traits, these quantities are grossly misinterpreted with such frequency as to warrant further comment [26]. Heritability is the fraction of variance in a phenotype that can be explained by genetic factors. Thus, heritability is not constant across environments, age, or physical location, as any changes in the variability of an external factor influencing a trait will necessarily induce corresponding changes in the heritability of said trait. Likewise, heritability is not an index of the immutability or malleability of a particular trait. A well-known hypothetical example aids our intuition here: if all humans were to smoke several packs of cigarettes a day throughout their lifespans but otherwise acted in the best interests of their physical health, there would be little variability in environmental factors related to the development of lung cancer, and the heritability of lung cancer would be nearly 100%. Of course, this says little of the malleability of risk factors for lung cancer; indeed, public health campaigns aimed at smoking cessation would likely greatly reduce lung cancer deaths in such a population. Thus, heritability is no more tied to genetic liability than it is to the environment in which it is measured.

1.4 The difficulty of genomic variance components analysis

With necessary precautions out of the way, we focus on the unique difficulties posed by the numerical estimation of genomic variance components models, again first considering the classical twin model to aid in our exposition.

The log-likelihood of the variance parameters θ governing the distribution of random vector

$y \sim \mathcal{MVN}(\vec{0}, \Sigma(\theta))$ is of the form

$$\ell(\theta|y) \propto \ln \det \Sigma(\theta) + y^T \Sigma(\theta)^{-1} y. \tag{1.4.1}$$

Here we assume $\Sigma(\theta) \in \mathbb{S}^n_{++}$ for all $\theta \in \Theta$, that is, for all parameter values θ belonging to the parameter space of interest Θ. Simply evaluating the likelihood, let alone finding $\theta^\star = \arg\max_\theta \ell(\theta|y)$, requires the expensive matrix operations involved in computing the log determinant $\ln \det \Sigma(\theta)$ and in solving the linear system $\Sigma(\theta)z = y$, both of which are $O(n^3)$ using direct methods. However, both the size and the structure typical in the classical twin model and the other problems which motivated the development of variance components analysis (e.g., unbalanced, hierarchical, and/or repeated-measures experimental design), rendered such operations feasible. For example, whereas samples consisting of more than 2,000 twin pairs are uncommon [21], samples consisting of hundreds of thousands of unrelated individuals are both common to and necessary for genomic variance components estimation and available sample sizes will continue to grow. Hence, in the case of genomic variance components models, estimation procedures with cubic time complexity are becoming exceedingly infeasible.

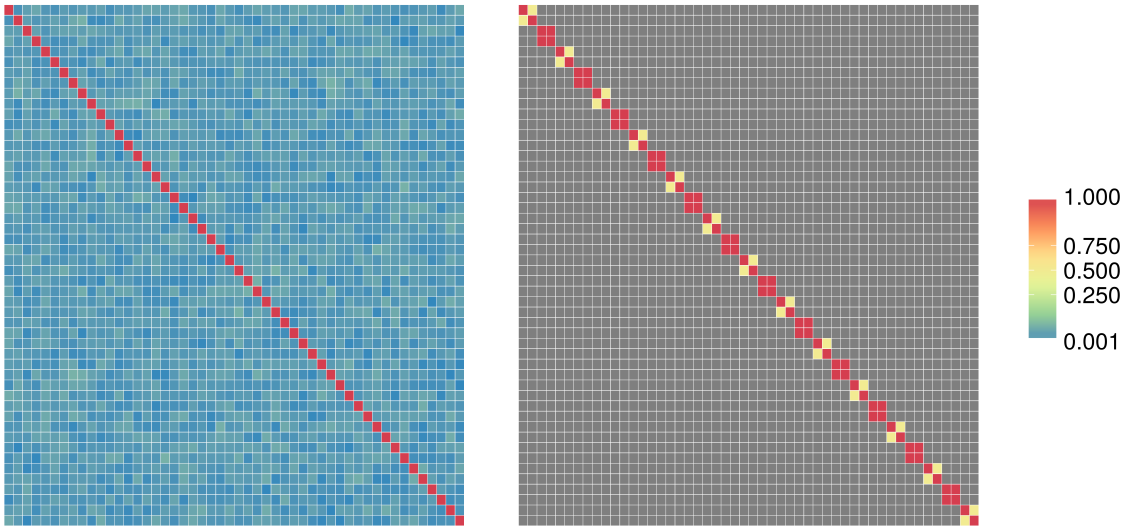
Greatly compounding the computational difficulties engendered by the large samples encountered in genomic applications is the lack of exploitable structure or sparsity of covariance matrices. Consider again the case of the classical twin model (1.3.2). Here, the covariance is of the form

$$\Sigma(\theta) = \sigma_a^2 A + \sigma_c^2 C + \sigma_e^2 I, \tag{1.4.2}$$

$$\begin{aligned}\{\Sigma(\theta)\}_{i,j} = \sigma_a^2 \, [\![i,j \text{ form an MZ pair}]\!] + \frac{\sigma_a^2}{2} \, [\![i,j \text{ form an DZ pair}]\!] = \\ + \sigma_c^2 \, [\![i,j \text{ form a sibling pair}]\!] + \sigma_e^2.\end{aligned} \tag{1.4.3}$$

Organizing siblings to be adjacent to one another then yields a block tridiagonal matrix with the

Figure 1.1: Density/sparsity of the genetic covariance structures in genomic variance components versus the classical twin model



The absolute value of the (dense) genomic correlation matrix (left) versus the (sparse, tridiagonal) A matrix in 1.3.2 (right). Grey indicates exactly zero cells.

following sparsity pattern:

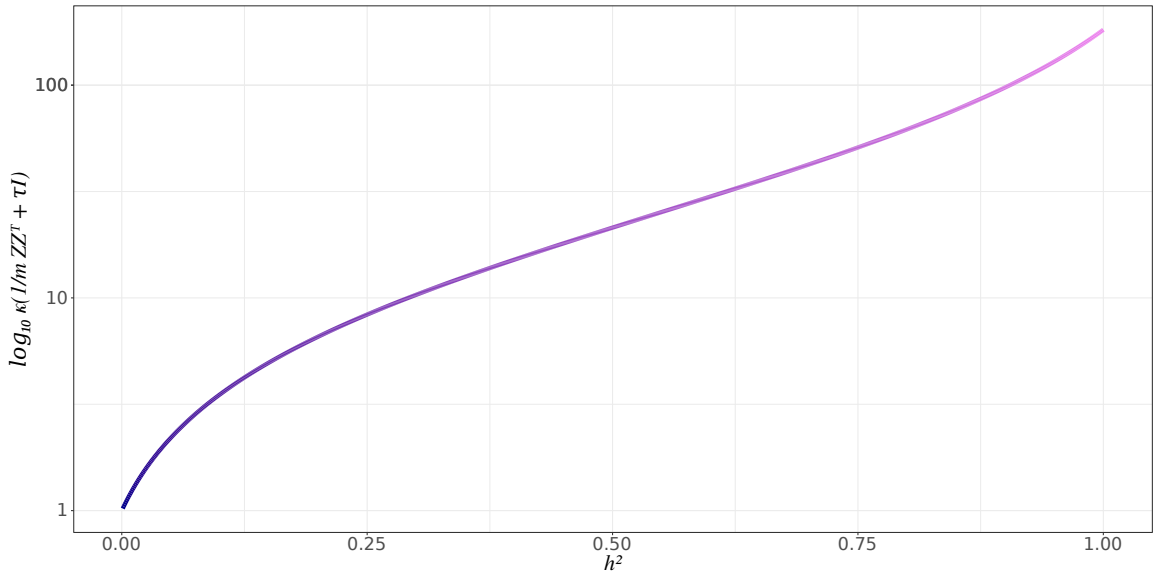
$$\begin{pmatrix} \times & \times & & & & & & \\ \times & \times & & & & & & \\ & & \times & \times & & & & \\ & & \times & \times & & & & \\ & & & & \ddots & \ddots & & \\ & & & & \ddots & \ddots & & \\ & & & & & & \times & \times \\ & & & & & & \times & \times \end{pmatrix} \tag{1.4.4}$$

Here, both the log determinant and the linear system have linear time complexity and evaluating the log likelihood would thus be inexpensive even for extremely large samples [9].

The GRM $\frac{1}{m}ZZ^T$, on the other hand, is not only large ($n \geqslant 100,000$) but dense, with all off-diagonal entries close to, but not equal to zero. Further, there is no natural ordering of the rows/columns, and, beyond that implied by positive definiteness, the matrix is unstructured. Figure 1.1 compares the structure of the GRM to the A matrix of the classical twin model.

Despite these difficulties, the overall covariance is the sum of a positive definite matrix and

Figure 1.2: Condition number of covariance matrix for varying h^2



The (spectral) condition number of $\frac{1}{m}ZZ^T + \tau I$ *where* $\Sigma(\theta)$ *in (1.4.5) is such that* $\Sigma(\theta) = \sigma_g^2\left(\frac{1}{m}ZZ^T + \tau I\right)$, $\tau = \sigma_e^2/\sigma_g^2$. *This choice of parameterization will be justified in 6.*

a scalar multiple of the identity:

$$\Sigma(\theta) = \sigma_g^2 \underbrace{\frac{1}{m}ZZ^T}_{\substack{\text{potentially} \\ \text{ill-cond.}}} + \underbrace{\sigma_e^2}_{>0}\underbrace{I_n}_{\substack{\text{well} \\ \text{cond.}}}, \tag{1.4.5}$$

Thus, $\Sigma(\theta)$ is generally well conditioned (assuming the phenotype of interest isn't $\approx 100\%$ heritable, which we can easily assume) and therefore amenable to iterative methods for matrix operations (Figure 1.2).

1.5 Previous work and our contributions

Previous approaches to mitigating the computational burden associated with genomic variance components likelihood estimation have generally taken one of two approaches:

1. limiting the number of $O(n^3)$ computations to the greatest extent possible;

2. substituting iterative and/or stochastic methods to avoid $O(n^3)$ computations all together.

For an example of the first approach, [15] and [29] compute the eigendecomposition of $\frac{1}{m}ZZ^T$ once and, after completing a few matrix-vector operations, employ a series of clever algebraic

manipulations to repeatedly evaluate the likelihood and its derivatives only using vector operations. For the second, [16] solve the necessary linear systems via the method of conjugate gradients and employ a Monte Carlo procedure to enable a series of computations that avoids evaluation of the log determinant and other trace computations.

The present thesis proposes methods in line with the second approach. However, in comparison with previous methods, we propose to exploit additional structure inherent to the problem of interest. As a result, the proposed algorithms will require only a single $O(n^2\eta)$ step, where η depends on $\tilde{\kappa} = \max_{\theta\in\Theta} \kappa\left(\Sigma(\theta)\right)$.

1.6 Outline of thesis

In the second chapter, we provide a brief introduction to variance component models and likelihood estimation, focusing on Newton-type estimation procedures. In the third chapter, we further discuss previous approaches to reducing the computational burden of genomic variance components estimation. The fourth and fifth chapters develop the theory of Krylov subspace methods and stochastic trace estimators, both of which are central to our proposed method. The sixth chapter describes the method in detail and includes the results of numerical experiments to support their effectiveness. The final chapter summarizes the present work in terms of its contributions and limitations and suggests directions for future research.

Chapter 2

Likelihood estimation of variance components

2.1 Introduction

In this chapter we develop a simple variance components model. We then derive the maximum likelihood (ML) objective function and its derivatives, as well as the best unbiased linear predictors (BLUPs) of the latent variables. Finally, we introduce Newton-type methods for ML and discuss several widely used Newton-type algorithms for solving the resulting optimization problems. We only consider simple models with no covariates and hence avoid discussion of maximum likelihood versus "restricted" or "residual" maximum likelihood estimators. Our presentation borrows heavily from [11] and [23].

2.2 Notation and general model

We are interested in models of the general form

$$(y|u,\theta) \sim \mathcal{MVN}\left(Zu, \sigma_e^2 R\right)$$
$$u \sim \mathcal{MVN}\left(\vec{0}, \sigma_g^2 G\right)$$

where the notation is as follows:

- $y \in \mathbb{R}^n$, the **outcome vector**, is an observed vector standardized to unit variance and zero mean.
- $Z \in \mathbb{R}^{n \times m}$, the **genotype matrix**, is an observed rank $\min(n, m)$ matrix of standardized genotypes assumed to have random effects. In our applications, each j^{th} column contains the standardized genotypes of n individuals at the j^{th} of m loci, with $m \gg n$.
- $u \in \mathbb{R}^m$, the **random effects**, are latent Gaussian random vectors with zero expectation.
- $G \in \mathbb{S}^m_{++}$ and $R \in \mathbb{S}^n_{++}$, the **random effect covariance structure** and the **residual covariance structure matrices**, respectively, are symmetric positive definite. In the cases considered later, these will simply be identity matrices.

- $\theta = \left(\sigma_e^2, \sigma_g^2\right)^T$, the **variance components**, is an unknown vector comprised of the **heritable variance component** σ_g^2 and the **non-heritable** or **residual variance component** σ_e^2.

For the sake of generality, however, we will often instead avoid explicit references to the variance components and instead move their influence into the matrices as G_θ, R_θ (or sometimes as $G(\theta)$, $R(\theta)$). For example, we might instead write the above model as

$$\begin{aligned} y|u &\sim \mathcal{MVN}\left(Zu, R_\theta\right) \\ u &\sim \mathcal{MVN}\left(\vec{0}, G_\theta\right), \end{aligned}$$

or, equivalently, in terms of the marginal distribution of y, as

$$y \sim \mathcal{MVN}\left(\vec{0}, R_\theta + ZG_\theta Z^T\right).$$

Alternatively, we may choose to represent the non-heritable variable component as a latent random variable $e \sim \mathcal{MVN}\left(Zu, R_\theta\right)$, in which case we write the model in generative form as

$$y = Zu + e, \quad u \sim \mathcal{MVN}\left(\vec{0}, G_\theta\right), \quad e \sim \mathcal{MVN}\left(\vec{0}, R_\theta\right)$$

In the models considered in the present thesis, we usually assume $R_\theta = \sigma_e^2 I_n$ and $R_g = \sigma_g^2 I_m$.

2.3 ML estimation of variance components

We are generally concerned with determining the values of the variance components θ that maximize the marginal log likelihood via Newton-type methods. To do so, we derive expressions for the value, gradient, and Hessian of the likelihood objective. Our derivations make repeated use of the facts presented in Appendix B.

Again consider the simple model

$$y = Zu + e, \quad u \sim \mathcal{MVN}\left(\vec{0}, G_\theta\right), \quad e \sim \mathcal{MVN}\left(\vec{0}, R_\theta\right),$$

which we write as $y \sim \mathcal{MVN}(\vec{0}, H)$, where $H = R_\theta + ZG_\theta Z^T$. The marginal log likelihood is

$$\ell_{ML}\left(\theta|y\right) \propto -\frac{1}{2}\left(\ln\left\{\det\left(H\right)\right\} + y^T H^{-1} y\right).$$

Letting $\dot{H}_j$ denote $\frac{\partial H}{\partial \theta_j}$, we derive the components of the score function $\mathcal{V}_{ML}(\theta) = \nabla \ell_{ML}\left(\theta|y\right)$ as:

$$\begin{aligned}
\frac{\partial \ell_{ML}}{\partial \sigma_j^2} &= -\frac{1}{2}\left(\text{Tr}\ \left(H^{-1}\dot{H}_j\right) - y^T H^{-1}\dot{H}_j H^{-1} y\right) \\
\implies \frac{\partial \ell_{ML}}{\partial \sigma_e^2} &= -\frac{1}{2}\left(\text{Tr}\ \left(H^{-1}R_\theta\right) - y^T H^{-1} R_\theta H^{-1} y\right), \\
\frac{\partial \ell_{ML}}{\partial \sigma_g^2} &= -\frac{1}{2}\left(\text{Tr}\ \left(H^{-1}ZG_\theta Z^T\right) - y^T H^{-1} ZG_\theta Z^T H^{-1} y\right)
\end{aligned}$$

The components of the **observed information matrix**, $\mathcal{I}_{ML}^{\text{obs}}(\theta) = -\nabla^2 \ell_{ML}$, which is the negative Hessian matrix of the log likelihood, are then derived

$$\begin{aligned}
\frac{\partial^2 \ell_{ML}}{\partial \sigma_i^2 \partial \sigma_j^2} &= -\frac{1}{2}\left(\text{Tr}\ \left(-H^{-1}\dot{H}_i H^{-1}\dot{H}_j\right) + 2y^T\left(H^{-1}\dot{H}_i H^{-1}\dot{H}_j H^{-1}\right) y\right) \\
\implies \frac{\partial^2 \ell_{ML}}{\partial \sigma_e^4} &= -\frac{1}{2}\left(\text{Tr}\ \left(-H^{-1}R_\theta H^{-1}R_\theta\right) + 2y^T\left(H^{-1}R_\theta H^{-1}R_\theta H^{-1}\right) y\right), \\
\frac{\partial^2 \ell_{ML}}{\partial \sigma_g^2 \partial \sigma_e^2} &= -\frac{1}{2}\left(\text{Tr}\ \left(-H^{-1}ZG_\theta Z^T H^{-1}R_\theta\right) + 2y^T\left(H^{-1}ZG_\theta Z^T H^{-1}R_\theta H^{-1}\right) y\right), \\
\frac{\partial^2 \ell_{ML}}{\partial \sigma_g^4} &= -\frac{1}{2}\left(\text{Tr}\ \left(-H^{-1}ZG_\theta Z^T H^{-1}ZG_\theta Z^T\right) + 2y^T\left(H^{-1}ZG_\theta Z^T H^{-1}ZG_\theta Z^T H^{-1}\right) y\right).
\end{aligned}$$

2.4 Prediction of latent variables

We briefly discuss equations for computing the Best Unbiased Linear Predictions (BLUPs) of the latent random variables for a given estimate of the unknown variable components, following the presentation of [23]. The distributions of the observed y, the latent u, and the residuals e can be characterized as

$$\begin{aligned}
(y|\theta) &\sim \mathcal{MVN}\left(\vec{0}, H\right) \text{ where } H = ZG_\theta Z^T + R_\theta, \\
(y|u,\theta) &\sim \mathcal{MVN}\left(Zu, R_\theta\right), \\
(u|\theta) &\sim \mathcal{MVN}\left(\vec{0}, G_\theta\right),
\end{aligned}$$

The joint log likelihood is factored as

$$\begin{aligned}\ell(y,u) =& \ell(y|u) + \ell(u) \\ &\propto -\left((y-Zu)^T R_\theta^{-1}(y-Zu) + u^T G_\theta^{-1} u + \ln\{\det\{R_\theta\}\} + \ln\{\det\{G_\theta\}\}\right) \\ &= -\left(y^T R_\theta^{-1} y - 2y^T R_\theta^{-1} Z u + u^T (G_\theta^{-1} + Z R_\theta^{-1} Z^T) u + \ln\{\det\{R_\theta\}\} + \ln\{\det\{G_\theta\}\}\right)\end{aligned}$$

The joint conditional log likelihood is maximized in u at the stationary point $\tilde{u}$ that solves the linear system:

$$(G_\theta^{-1} + Z^T R_\theta^{-1} Z)\tilde{u} = Z^T R_\theta^{-1} y$$

The above corresponds to a simple case of what are commonly referred to as "Henderson's mixed model equations", and its solution, $\tilde{u} = G_\theta Z^T H_\theta^{-1} y$ provides the BLUPs for the latent genetic effects. If we represent the residual variation in y by the latent variable e (such that $y = Zu + e$, $e \sim \mathcal{MVN}(\vec{0}, R_\theta)$), we can immediately derive the BLUPs for e given θ as

$$\begin{aligned}\tilde{e} &= y - Z\tilde{u} \\ &= \left(H_\theta - Z G_\theta Z^T\right) H_\theta^{-1} y \\ &= R_\theta H_\theta^{-1} y.\end{aligned}$$

2.5 Newton-type algorithms for ML estimation

We consider Newton-type methods for likelihood estimation of variance components; that is, determining the values of the variance components θ that maximize the marginal log likelihood. The procedures introduced only differ in their choices of the negative Hessian-like information matrix employed in the Newton iteration. A general template for Newton-type algorithms for ML estimation is presented in Algorithm 2.1. We denote the genomic relatedness matrix as $K = \frac{1}{m} Z Z^T$. We define three Newton-type methods based on the Hessian-type matrix used:

The **Newton Raphson** algorithm uses the observed information matrices derived in the previous sections:

$$\left\{\mathcal{I}_{ML}^{\text{obs}}(\theta)\right\}_{ij} = \frac{1}{2}\left\{\text{Tr}\,\left(-H^{-1}\dot{H}_i H^{-1}\dot{H}_j\right) + 2y^T\left(H^{-1}\dot{H}_i H^{-1}\dot{H}_j H^{-1}\right) y\right\}_{ij},$$

Algorithm 2.1 General Newton-type algorithm

input: starting value $\theta_0 \in \Theta$, score function $\mathcal{V}$, information function $\mathcal{I}$, convergence criterion `cvrg`
initialize: $j = 0$, $\theta_{-1} = \infty$
1: **while** not $\texttt{cvrg}(\theta_j, \theta_{j-1}, j)$ **do**
2: $\quad \theta_{j+1} = \theta_j + \mathcal{I}^{-1}(\theta_j)\mathcal{V}(\theta_j)$
3: $\quad j \leftarrow j + 1$
4: **end while**
5: **return** estimated variance components θ_j

An alternative approach known as **Fisher Scoring** substitutes the expected value (with respect to the outcome variable) of the Hessian of the log likelihood for the observed Hessian. The negative expected Hessian is the so called **Fisher Information Matrix**, which is given by

$$\begin{aligned}\left\{\mathcal{I}^{\mathbb{E}}_{ML}(\theta)\right\}_{ij} =& \frac{1}{2}\left\{\mathbb{E}_{y|\theta}\left[\mathrm{Tr}\,\left(-H^{-1}\dot{H}_i H^{-1}\dot{H}_j\right) + 2y^T\left(H^{-1}\dot{H}_i H^{-1}\dot{H}_j H^{-1}\right)y\right]\right\}_{ij} \\ =& \frac{1}{2}\left\{\mathrm{Tr}\,\left(-H^{-1}\dot{H}_i H^{-1}\dot{H}_j\right)\right\}_{ij} + 2\mathrm{Tr}\,\left(H^{-1}\dot{H}_i H^{-1}\dot{H}_j H^{-1}H\right) + 0 \\ =& \frac{1}{2}\left\{\mathrm{Tr}\,\left(H^{-1}\dot{H}_i H^{-1}\dot{H}_j\right)\right\}_{ij}\end{aligned}$$

as $\mathbb{E}_{y|\theta}[y] = 0$. Finally, the popular **Average Information** algorithm uses the average of the observed and expected information matrices. It has been shown that this procedure is optimal in the sense that the information matrix can be split into two components, one of which is a random matrix with zero-expectation; the average information excludes this uninformative random matrix [30]. The average information matrix is given by

$$\left\{\mathcal{I}^{\mathrm{AI}}_{ML}(\theta)\right\}_{ij} = \frac{1}{2}\left(\left\{\mathcal{I}^{\mathrm{obs}}_{ML}(\theta)\right\}_{ij} + \left\{\mathcal{I}^{\mathbb{E}}_{ML}(\theta)\right\}_{ij}\right) = 2y^T\left(H^{-1}\dot{H}_i H^{-1}\dot{H}_j H^{-1}\right)y.$$

2.6 Looking ahead

With suitable chosen starting values, the above algorithms are effective for small to moderately sized problems (i.e., $n \lessapprox 50,000$). However, naive implementations using direct methods for matrix operations quickly become infeasible as n grows large. In the next chapter, we review a selection of previously proposed approaches to computationally efficient estimation of genomic variance components.

Chapter 3

Optimizing performance of estimation algorithms

3.1 Overview

Variance components estimation algorithms, including those presented in the previous chapter, work well for moderately sized data sets, particularly when sparsity or other structure in the variance/covariance matrix can be exploited. As we argued in the introductory chapter, the estimation of genomic variance component models enjoys neither moderate sample sizes nor simple covariance structures. Many software packages aiming to address these difficulties have been developed over the previous eight years. Here we present two quintessential approaches. The first approach, which involves a single $O(n^3)$ operation, is extremely effective in certain scenarios, but is fundamentally limited in other respects. The second approach, which employs stochastic and iterative procedures, forms a foundation for the novel algorithms we later propose.

3.2 Spectral decomposition based Newton Raphson

The Newton-type methods described in the previous section involve multiple $O(n^3)$ matrix operations. For a limited class of genomic variance components models, clever algebraic manipulations can accomplish ML/REML variance component estimation performing only a single $O(n^3)$ operation: computing the spectral decomposition of the GRM $\frac{1}{m}ZZ^T$. Below we present such a version of Average Information Newton Raphson based on those of [15, 29] for the model parameterized

$$y \sim \mathcal{MVN}\left(X\beta, \tau^{-1}(\gamma K + I_n)\right).$$

Here K is the GRM $K = \frac{1}{m}ZZ^T$ and the variance components have been parameterized as $\tau = 1/\sigma_e^2$, and $\gamma = \sigma_g^2/\sigma_e^2$. For simplicity of presentation, we assume no fixed effects (i.e., $X\beta = 0$), though this is not an inherent limitation of the method.

We seek to efficiently compute Newton updates of the form

$$(\tau, \gamma)^T \leftarrow (\tau, \gamma)^T + \mathcal{I}_{AI}^{-1}(\tau, \gamma)\mathcal{V}(\tau, \gamma)$$

Recall that the Average Information matrix, for which we must solve the above linear system at each Newton iteration, is given by

$$\mathcal{I}_{AI}(\tau, \gamma) = \frac{1}{4}\begin{bmatrix} 2n\tau^{-2} & -\tau^{-1}\mathrm{Tr}\left(V^{-1}K\right) - y^T V^{-1}KV^{-1}y \\ -\tau^{-1}\mathrm{Tr}\left(V^{-1}K\right) - y^T V^{-1}KV^{-1}y & 2\tau y^T V^{-1}KV^{-1}KV^{-1}y \end{bmatrix},$$

and the score function and likelihood are

$$\mathcal{V}(\tau, \gamma) = -\frac{1}{2}\begin{bmatrix} -N\tau^{-1} + y^T V^{-1}y \\ \mathrm{Tr}\ \left(V^{-1}K\right) - \tau y^T V^{-1}KV^{-1}y \end{bmatrix},$$
$$\ell\left(\tau, \gamma | y\right) = -\frac{1}{2}\left(-n\ln\left(\tau\right) + \ln\left(\det V\right) + \tau y^T V^{-1}y + n\ln(2\pi)\right),$$

where $V = \gamma K + I_n$. Denote the eigendecomposition of K as

$$K = U\Lambda U^T.$$

Evaluation of the objective function, its gradient, and its inverse Hessian require computing the following expressions:

$$\mathrm{Tr}\ \left(V^{-1}K\right), \quad \det V,$$
$$y^T V^{-1}y, \quad y^T V^{-1}KV^{-1}y, \quad y^T V^{-1}KV^{-1}KV^{-1}y.$$

The trace term can be obtained in $O(n)$ by

$$\begin{aligned} \mathrm{Tr}\ \left(V^{-1}K\right) &= \mathrm{Tr}\ \left(\left(\gamma U\Lambda U^T + UU^T\right)^{-1} U\Lambda U^T\right) \\ &= \mathrm{Tr}\ \left(U\left(\gamma\Lambda + I_n\right)^{-1} U^T U\Lambda U^T\right) \\ &= \mathrm{Tr}\ \left(\left(\gamma\Lambda + I_n\right)^{-1}\Lambda\right) \\ &= \mathrm{Tr}\ \left(\mathrm{diag}\left(\frac{1}{\gamma\lambda_i + 1}\right)\Lambda\right) = \sum_i \left(\frac{\lambda_i}{\gamma\lambda_i + 1}\right), \end{aligned}$$

where $\lambda_1, \ldots, \lambda_n$ denote the diagonal entries of Λ (i.e., the eigenvalues of K). Similarly, the determinant can be obtained by

$$\det V = \det\left(U\left(\gamma\Lambda + I_n\right)U^T\right) = \det\left(\gamma\Lambda + I_n\right) = \prod_i \left(\gamma\lambda_i + 1\right).$$

The inner-product terms are obtained:

$$\begin{aligned}
y^T V^{-1} y &= y^T U \left(\gamma\Lambda + I_n\right)^{-1} U^T y, \\
y^T V^{-1} K V^{-1} y &= y^T U \left(\gamma\Lambda + I_n\right)^{-1} U^T K U \left(\gamma\Lambda + I_n\right)^{-1} U^T y \\
&= y^T U \left(\gamma\Lambda + I_n\right)^{-1} U^T U \Lambda U^T U \left(\gamma\Lambda + I_n\right)^{-1} U^T y \\
&= y^T U \left(\gamma\Lambda + I_n\right)^{-1} \Lambda \left(\gamma\Lambda + I_n\right)^{-1} U^T y, \\
y^T V^{-1} K V^{-1} K V^{-1} y &= y^T U \left(\gamma\Lambda + I_n\right)^{-1} \Lambda \left(\gamma\Lambda + I_n\right)^{-1} \Lambda \left(\gamma\Lambda + I_n\right)^{-1} U^T y.
\end{aligned}$$

As all terms in between the matrix-vector products $\psi^T = y^T U$ are products of diagonal matrices, these computations are all $O(n)$:

$$\begin{aligned}
y^T V^{-1} y &= \sum_i \frac{\psi_i^2}{\gamma\lambda_i + 1}, \\
y^T V^{-1} K V^{-1} y &= \sum_i \frac{\psi_i^2 \lambda_i}{\left(\gamma\lambda_i + 1\right)^2}, \\
y^T V^{-1} K V^{-1} K V^{-1} y &= \sum_i \frac{\psi_i^2 \lambda_i^2}{\left(\gamma\lambda_i + 1\right)^3}.
\end{aligned}$$

Thus, provided we are willing to compute the eigendecomposition of the GRM, after computing the inner products of the phenotype vector with the eigenvectors of the GRM, the entire average information Newton Raphson algorithm can be implemented using only vector operations. This may be particularly advantageous when variance component estimation is desired for multiple phenotypes $y_1, \ldots, y_L$ measured in the same group of n individuals. The eigendecomposition $U\Lambda U^T$ may then be reused, and NR can again be performed in $O(n)$ after computing L matrix vector products of the form $U^T y_\ell$. We implement this method in Appendix C.2.

There are, however, two fundamental limitations to this technique. First, though it may be

extended to variance structures of the form

$$V = \gamma K + R, \quad R \in \mathbb{S}^n_{++},$$

by solving the generalized eigenvalue problem for the matrix pencil (K, R), it is not in general possible to accommodate three or more matrices

$$V = \gamma_1 K_2 + \gamma_2 K_2 + R,$$

as arise in partitioned heritability analysis[1]. Further, even for the simpler case presented above, *any* $O(n^3)$ computations will become prohibitively expensive for large n.

3.3 Monte Carlo Newton Raphson

An alternative approach, due to [7], is the Monte Carlo Newton Raphson (MCNR) method. Here we present the implementation of [18, 16] that pairs MCNR with iterative matrix methods to avoid all $O(n^3)$ computations. Compared to the standard NR methods of the previous chapter, MCNR avoids explicit evaluation of the likelihood or its gradients. Again, we consider a simplified model for clarity, albeit with a slightly different parameterization, which we present in marginal

[1] For matrices $A_1, A_2, A_3 \in \mathbb{S}^n_{++}$, suppose there exists unitary U that simultaneously diagonalize all three matrices:

$$UA_1U^T = \Lambda_1, \quad UA_2U^T = \Lambda_2, \quad UA_3U^T = \Lambda_3.$$

For the first two matrices, this is equivalent to the eigenvalue problem for $A' \equiv A_2A_1^{-1}$ as we have

$$\begin{aligned} U &= A_1^{-1}U\Lambda_1 \\ \implies A_1^{-1}U\Lambda_1 &= A_2^{-1}U\Lambda_2 \\ \implies U^T\underbrace{A_2A_1^{-1}}_{\equiv A'}U &= \underbrace{\Lambda_1\Lambda_2^{-1}}_{\equiv \Lambda'}. \end{aligned}$$

For all three matrices, however, the same argument yields

$$\begin{aligned} U^TA_2A_1^{-1}U &= \Lambda_1\Lambda_2^{-1} \\ U^TA_3A_1^{-1}U &= \Lambda_1\Lambda_3^{-1} \\ U^TA_3A_2^{-1}U &= \Lambda_2\Lambda_3^{-1} \end{aligned}$$

and thus

$$\begin{aligned} U^TA_3A_2^{-1}A_3A_2^{-1}U = \Lambda_1\Lambda_3^{-1} &= U^TA_3A_1^{-1}U \\ \implies A_3A_2^{-1}A_3A_2^{-1} &= A_3A_1^{-1} \\ \implies A_2^{-1}A_3A_2^{-1} &= A_1^{-1}, \end{aligned}$$

which is, in general, an untenable assumption.

and generative form:

$$y \sim \mathcal{MVN}(\vec{0}, H^{-1}y), \quad H = \sigma_e^2 I_n + \sigma_g^2 \frac{1}{m} ZZ^T,$$
$$y = m^{-1/2} Zu + e, \quad u \sim \mathcal{MVN}(\vec{0}, \sigma_g^2 I_m), \quad e \sim \mathcal{MVN}(\vec{0}, \sigma_e^2 I_n).$$

Our presentation borrows from that of [20], [23], and [16].

Recall the first order conditions on the score function (i.e., setting the gradient of the likelihood to zero) presented in Section 2.3, which for the present model can be expressed as

$$\text{Tr } \left(H^{-1}K\right) = y^T H^{-1} K H^{-1} y,$$
$$\text{Tr } \left(H^{-1}\right) = y^T H^{-1} H^{-1} y,$$

where we have denoted the GRM $K = \frac{1}{m} ZZ^T$. For fixed estimates of the unknown variance components $\hat{\sigma}_e^2$, $\hat{\sigma}_g^2$, the BLUPs for the random u and e can be obtained by solving Henderson's MMEs and are given by $\tilde{u} = \hat{\sigma}_g^{-2} m^{-1/2} Z^T H^{-1} y$, $\tilde{e} = \hat{\sigma}_e^{-2} H^{-1} y$. Thus we have that $\hat{\sigma}_g^{-4} \tilde{u}^T \tilde{u} = y^T H^{-1} K H^{-1} y$, and $\hat{\sigma}_e^{-4} \tilde{e}^T \tilde{e} = y^T H^{-1} H^{-1} y$. Further, we compute the expectations

$$\hat{\sigma}_g^{-4} \mathbb{E}\left[u^T u | y\right] = \text{Tr } \left(H^{-1} K H^{-1} \mathbb{E}\left[yy^T\right]\right) = \text{Tr } \left(H^{-1} K H^{-1} H\right) = \text{Tr}\left(H^{-1}K\right), \tag{3.3.1}$$
$$\hat{\sigma}_e^{-4} \mathbb{E}\left[e^T e | y\right] = \text{Tr } \left(H^{-1} H^{-1} \mathbb{E}\left[yy^T\right]\right) = \text{Tr } \left(H^{-1}\right). \tag{3.3.2}$$

Thus, the first order condition $\nabla \ell(\sigma_e^2, \sigma_g^2 | y) = \vec{0}$ can be represented by the equations

$$\mathbb{E}\left[u^T u | y\right] = \tilde{u}^T \tilde{u},$$
$$\mathbb{E}\left[e^T e | y\right] = \tilde{e}^T \tilde{e}.$$

A necessary condition for the above is given by

$$\frac{\mathbb{E}\left[u^T u | y\right]}{\mathbb{E}\left[e^T e | y\right]} = \frac{\tilde{u}^T \tilde{u}}{\tilde{e}^T \tilde{e}}. \tag{3.3.3}$$

Loh et al. [18, 16] demonstrate that, in practice, choosing σ_e^2, σ_g^2 satisfying (3.3.3) is enough to

satisfy the first order conditions, which they achieve via finding the zero of the criterion function

$$f_{\text{crit}}(\sigma_e^2, \sigma_g^2) = \ln\left(\frac{\tilde{u}^T\tilde{u}}{\tilde{e}^T\tilde{e}} \cdot \frac{\mathbb{E}\left[e^T e|y\right]}{\mathbb{E}\left[u^T u|y\right]}\right),$$

where the expectations are approximated via a stochastic estimator. Specifically, making the change of variables $\tau = \frac{\sigma_e^2}{\sigma_g^2}$, $\gamma = \sigma_g^2$, such that $H = \gamma\tilde{H}$ with $\tilde{H} = \tau I_n + \frac{1}{m}ZZ^T$, we proceed as follows. For a given estimate of the variance components $\hat{\tau}$, $\hat{\gamma}$, we compute BLUPs

$$\tilde{u} = m^{-1/2}Z^T\tilde{H}^{-1}y, \quad \tilde{e} = \hat{\tau}\tilde{H}^{-1}y.$$

We then draw L random samples

$$\check{u}_\ell \sim \mathcal{MVN}(0, I_m), \quad \check{e}_\ell \sim \mathcal{MVN}(0, I_n), \quad \text{for } \ell = 1, \ldots, L,$$

and construct random phenotype vectors implied by the estimated variance components:

$$\check{y}_\ell = m^{-1/2}Z\check{u}_\ell + \hat{\tau}^{-1/2}\check{e}_\ell, \quad \text{for } \ell = 1, \ldots, L.$$

BLUPs are then computed for each of the random samples

$$\tilde{\check{u}}_\ell = m^{-1/2}Z^T\tilde{H}^{-1}\check{y}_\ell, \quad \tilde{\check{e}}_\ell = \hat{\tau}\tilde{H}^{-1}\check{y}_\ell,$$

and expectations are approximated by:

$$\mathbb{E}\left[u^T u|y\right] \approx \mathbb{E}_{\text{MC}}\left[u^T u|y\right] = \frac{1}{L}\sum_{\ell=1}^{L}\left\|\tilde{\check{u}}_\ell\right\|^2,$$
$$\mathbb{E}\left[e^T e|y\right] \approx \mathbb{E}_{\text{MC}}\left[e^T e|y\right] = \frac{1}{L}\sum_{\ell=1}^{L}\left\|\tilde{\check{e}}_\ell\right\|^2.$$

The variance components are then updated via

$$\tau \leftarrow \text{zero}_\tau f_{\text{MC}}(\tau), \quad f_{\text{MC}}(\tau) = \ln\left(\frac{\|\tilde{e}_\ell\|}{\|\tilde{u}_\ell\|} \cdot \frac{\mathbb{E}_{\text{MC}}\left[e^T e|y\right]}{\mathbb{E}_{\text{MC}}\left[u^T u|y\right]}\right), \tag{3.3.4}$$

until convergence, with the same random vectors $\check{u}_\ell, \check{e}_\ell$, $\ell = 1, \ldots, L$, reused at each iteration. We provide an implementation of this procedure in Section C.3, which, mirroring that of [16], uses the method of Conjugate Gradients to compute BLUPs and the secant method to find the roots of f_{MC} in 3.3.4.

3.4 Improving existing methods

The implementation of MCNR in [18] is, to our knowledge, the fastest widely-used software for estimating the simple two-component model:

$$y \sim \mathcal{MVN}(\vec{0}, H^{-1}y), \quad H = \sigma_e^2 I_n + \sigma_g^2 \frac{1}{m} Z Z^T.$$

The speed of the method is largely due to two factors: 1, using the iterative Conjugate Gradients method to solve dense linear systems, and 2, the introduction of a stochastic procedure for what is essentially the trace of a function of a matrix, as can be seen in equations (3.3.1)-(3.3.2).

In the next two chapters, we further develop the theory of Krylov subspace methods such as Conjugate Gradients and stochastic estimation of Tr $f(A)$ for $f : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$. In the process, we demonstrate that properties of these techniques are particularly well suited to the problem structure of genetics variance components estimation, forming the foundation for the algorithms we propose in Chapter 6.

Chapter 4

Krylov subspace methods

4.1 Overview

With the exception of the spectral decomposition based Newton Raphson algorithm (Section 3.2), which is inherently $O(n^3)$, all the algorithms considered thus far involve solving a linear system of the form

$$Ax = b, \quad A \in \mathbb{S}^n_{++}, \tag{4.1.1}$$

where, in the simplest case, A is a conic combination of a dense, high rank, genomic relatedness matrix and a diagonal matrix. Throughout the remainder of the thesis, we will consider this case, further simplified by assuming no fixed effects, parameterized as

$$\gamma A = \gamma \left(\tau I + \frac{1}{m} Z Z^T \right), \tag{4.1.2}$$

where $\tau = \sigma_e^2/\sigma_g^2$ is the ratio of the non-heritable and heritable variance components and $\gamma = \sigma_g^2$ is the heritable variance component. A comprises the covariance matrix of the random vector $y = b$[1] with marginal distribution

$$b \sim \mathcal{MVN}\left(\vec{0}, \gamma A\right).$$

As demonstrated in Chapter 2, the solution of a linear system involving A is necessary for evaluating the log likelihood and its derivatives or performing best unbiased linear prediction of the latent random effects.

Noting that τ is strictly positive, A is in general well-conditioned, with conditioning improving as the non-heritable component σ_e^2 (and hence τ) increases. This, combined with the fact that A is necessarily positive definite[2], makes the linear system $Ax = b$ amenable to iterative meth-

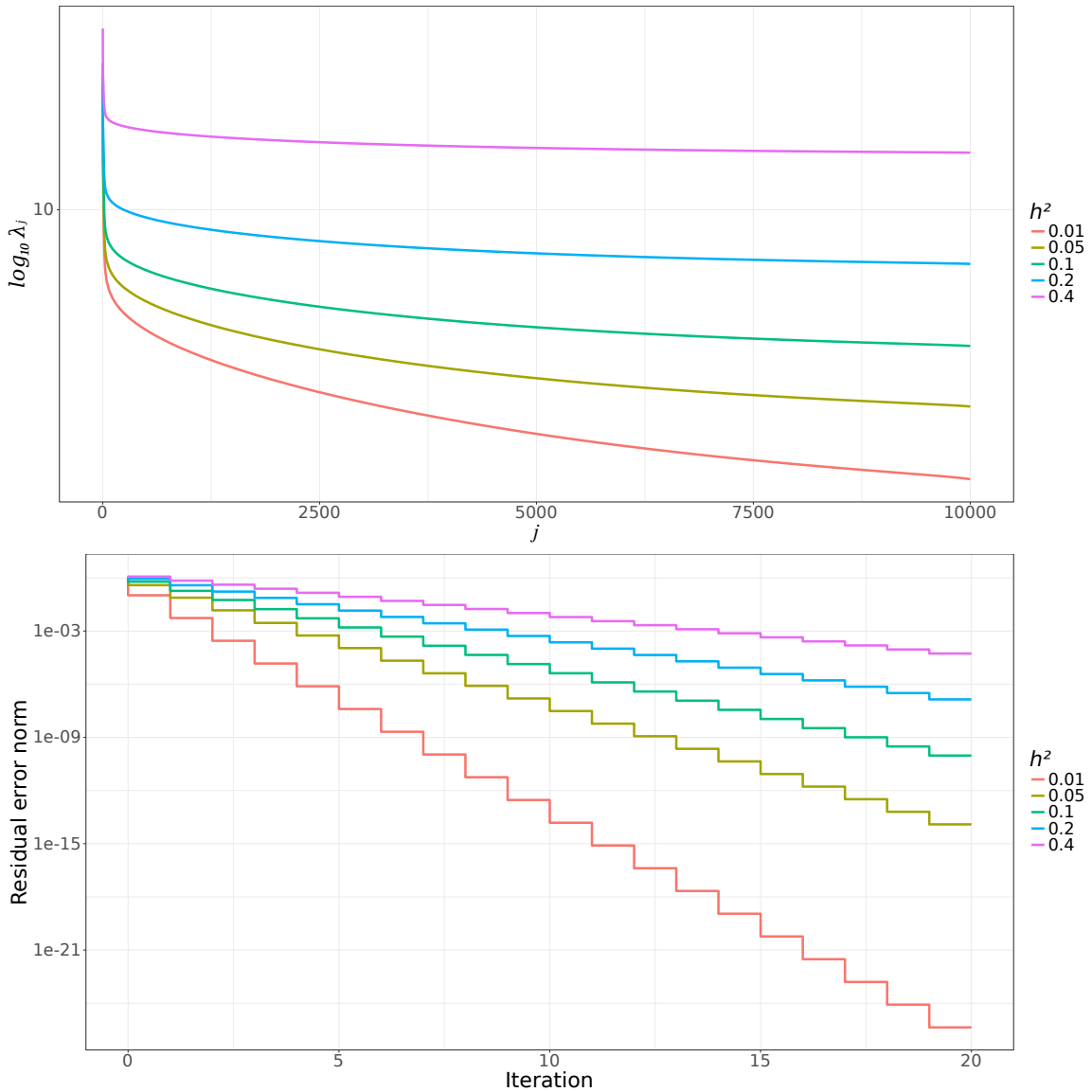
[1] We substitute "b" for "y" for the sake of congruence with the numerical linear algebra literature.

[2] To see this, note that even if we drop the assumption that Z is of full row rank, ZZ^T is a correlation matrix and hence $ZZ^T \in \mathbb{S}^T_+$. Denoting the spectral decomposition $ZZ^T = UDU^T$, we can write $A = U(D + \tau I)U^T = U\Lambda U^T$ with eigenvalues $\Lambda_{i,i} = d_{i,j} + \tau \geq 0 + \tau > 0$. Hence, $A \in \mathbb{S}^n_{++}$.

ods, particularly Krylov subspace methods (Figure 4.1). [16] use Conjugate Gradients to compute BLUPs for the latent genetic effects and residuals, which, combined with MC Newton Raphson (3.3), leads to an efficient genomic variance components estimation procedure.

The present chapter introduces the basic theory underlying Krylov subspace methods, then describes two extensions to standard Conjugate Gradients that are particularly well suited to the genomics variance components estimation problem. Finally, we present the results of numerical experiments illustrating the utility of these extensions.

Figure 4.1: Spectrum of covariance matrix and convergence of conjugate gradients for varying h^2



Top: the spectrum of $A = \frac{1}{m}ZZ^T + \tau I$ *for varying* h^2 *where* $\tau = (1-h^2)/h^2$. *Bottom: the decrease in the residual error norm, as a fraction of* $\|b\|_A$ *when solving* $Ax = b$ *via Conjugate Gradients.*

4.2 The Lanczos process

Here we briefly introduce Krylov subspace theory and the Lanczos process. We assume familiarity with the method of Conjugate Gradients (CG; (4.1)). Our objective is to demonstrate that the Lanczos process can be used to compute the CG successive approximations to the solution of $Ax = b$ for $A \in \mathbb{S}^n_{++}$ For simplicity, we discuss the general problem of finding x such that $Ax = b$. Much of our presentation follows that of [3] and omitted proofs can be found in [3] §4.2 - 4.3.

Algorithm 4.1 Conjugate Gradients (CG)

input: a matrix $A \in \mathbb{S}^n_{++}$, RHS $b \in \mathbb{R}^n_{\neq 0}$, and error tolerance $\epsilon > 0$
initialize: $x_0 = 0$, $p_0 = r_0 = b$, $i = 0$
repeat
  $\alpha_i = \frac{\langle r_i, r_i \rangle_I}{\langle p_i, p_i \rangle_A}$
  $x_{i+1} = x_i + \alpha_i p_i$
  $r_{i+1} = r_i - \alpha_i A p_i$
  $\beta_i = \frac{\langle r_{i+1}, r_{i+1} \rangle_I}{\langle r_i, r_i \rangle_I}$
  $p_{i+1} = r_{i+1} + \beta_i p_i$
  $i \leftarrow i + 1$
until $\|r_i\| \leqslant \epsilon$
return x_i

Definition 4.1. Given an initial approximation x_0 to the solution of $Ax = b$, define the initial residual $r_0 = b - Ax_0$. For $m = 1, \ldots, n$ define the m^{th} *Krylov subspace* by

$$\mathcal{K}_m(A, r_0) = \text{span}\{r_0, Ar_0, A^2 r_0, \ldots, A^{m-1} r_0\}.$$

Note that Krylov subspaces are nested by construction. That is,

$$\mathcal{K}_0 \subset \mathcal{K}_1 \subset \cdots \subset \mathcal{K}_{m-1} \subset \mathcal{K}_m, \ldots m = 1, \ldots, n,$$

where n is the rank of the (square) matrix A. For $A \in \mathbb{S}^n_{++}$ and $r_0 \neq 0$, the *Lanczos process* (4.2), defines a method for the sequential construction of an orthonormal basis $\{u_1, \ldots, u_m\}$ for $\mathcal{K}_m(A, r_0)$ such that $\text{span}\{u_1, \ldots, u_j\} = \mathcal{K}_j(A, r_0), \quad j = 1, \ldots, m$. Further details can be found in [3] §4.3.1.

Algorithm 4.2 The Lanczos process

input: $A \in \mathbb{S}^m_{++}$, $s_0 \in \mathbb{R}^m_{\neq 0}$

initialize: $u_0 = s_0/\|s_0\|$, $u_{-1} = 0$

1: **for** $j = 1, \ldots, m$ **do**

2: define $T \in \mathbb{R}^{j \times j}$ by

$$t_{i,j} = \begin{cases} \langle u_i, u_j \rangle_A & \text{if } i \leq j, \\ \left\| Au_j - \sum_{k=j-2}^{j} u_k t_{k,j} \right\|_2 & \text{if } i = j+1. \end{cases}$$

3: $\tilde{\beta}_j = t^{-1}_{j+1,j}$

4: $u_{j+1} = \tilde{\beta}_j \left(Au_j - \sum_{k=j-2}^{j} u_k t_{k,j} \right)$

5: **end for**

6: **return** orthogonal U, tridiagonal T such that $AU = UT$, col $U = \mathcal{K}_m(A, r_0)$

Though the Lanczos process is typically thought of in relation to the eigenvalue problem, Lanczos demonstrated that the process can be used to solve symmetric positive definite systems [13, 3]. Its equivalence to CG is demonstrated as follows.

The Lanczos process constructs the unique orthonormal basis $U_j = (u_1, \ldots, u_j)$ for $\mathcal{K}_j(A, b)$, where $u_1 = b/\|b\|$, and a tridiagonal matrix T_j whose entries we denote

$$T_j = \begin{pmatrix} \tilde{\alpha}_1 & \tilde{\beta}_2 & & \\ \tilde{\beta}_2 & \ddots & \ddots & \\ & \ddots & \ddots & \tilde{\beta}_j \\ & & \tilde{\beta}_j & \tilde{\alpha}_j \end{pmatrix},$$

such that $AU_j = U_j T_j$. We can write the j^{th} conjugate gradients approximation $x_j \in \mathcal{K}_j(A, b) = \text{col } U_j$ in terms of the basis expansion $x_j = U_j y$. By the optimality of CG,

$$x_j = \min_{x \in \mathcal{K}_j(A,b)} x^T Ax - x^T b,$$

where the right hand side simplifies as

$$\begin{aligned} x^T Ax - x^T b =& y_j^T U_j^T A U_j y_j - y_j^T U_j^T b \\ =& y_j^T T_j y_j - \|b\| y_j^T e_1. \end{aligned}$$

The above quadratic form is minimized at $y_j = T_j^{-1}\|b\|e_1$, thus providing the link between CG and

the Lanczos process. Explicit formulae for deriving the classical CG coefficients from the Lanczos procedure, which only requires vector operations, are given in [6, 3]. We present Frommer et al.'s algorithm, which we term Lanczos Conjugate Gradients (LCG), in 4.3.

Algorithm 4.3 Lanczos Conjugate Gradients (LCG; [6])

input: a matrix $A \in \mathbb{S}^m_{++}$, RHS $b \in \mathbb{R}^m_{\neq 0}$, error tolerance $\epsilon > 0$
initialize: $x_0 = 0$, $p_0 = r_0 = b$, $\omega_{-1} = 0$, $\gamma_{-1} = 1$, $u_{-1} = 0$, $u_0 = r_0/\beta_0$, $i = 0$
repeat
  compute next orthonormal Krylov basis vector $u_{j+1} \perp u_{\ell<j+1}$:
    $\delta_i = \langle u_i u_i \rangle_A$
    $\tilde{u}_{j+1} = Au_i - \delta_i u_i, -\beta_i u_i$
    $\beta_{i+1} = \|\tilde{u}_{j+1}\|$
    $u_{j+1} = \tilde{u}_{j+1}/\beta_{j+1}$
  update approximate solution, search direction, and residual as in CG:
    $\gamma_i = (\delta_i - \omega_{i-1}/\gamma_{i-1})^{-1}$
    $\omega_i = (\beta_{i+1}\gamma_i)^2$
    $\rho_{i+1} = -\beta_{i+1}\rho_i$
    $x_{i+1} = x_i + \gamma_i p_i$
    $r_{i+1} = \rho_{j+1} u_{j+1}$
    $p_{i+1} = r_{i+1} + \omega_i p_i$
  $i \leftarrow i + 1$
until $\|r_i\| \leqslant \epsilon$
return x_i

4.3 Block Lanczos Conjugate Gradients (BLCG)

Here we introduce a straightforward extension of Krylov subspace methods for solving linear systems with multiple right hand sides:

$$Ax_i = b_i, \quad i = 1, \ldots, t, \quad A \in \mathbb{S}^n_{++} \tag{4.3.1}$$

An equivalent problem, using matrix notation, is that of finding $X = [x_1, \ldots, x_t]$ such that $AX = B$ where $B = [b_1, \ldots, b_t]$. We can use the *Block Lanczos procedure,* following [2], to construct orthonormal bases for the m^{th} *block Krylov subspace*, defined by

$$\begin{aligned}\mathcal{K}_m(A, B) &= \text{span}\{b_1, \ldots, b_t, Ab_1, \ldots, Ab_t, A^2b_1, \ldots A^{m-1}b_1, \ldots, A^{m-1}r_t\} \\ &= \text{col}\left[B, AB, A^2B, \ldots, A^{m-1}B\right].\end{aligned}$$

For each column b_i of B, this process yields the familiar Lanczos decomposition components U, T, orthonormal and tridiagonal, respectively, such that $AT = TU$ and col $U = \mathcal{K}_m(A, b_i)$. Again, CG iterates can be computed with $O(n^2)$ operations for each of the linear systems in (4.3.1). We present a simple procedure, which we term Block Lanczos Conjugate Gradients (BLCG), in Algorithm 4.4 and provide an implementation in Appendix C.4. We note that, in comparison to performing LCG on each of the systems, $Ax_1 = b_1, \ldots, Ax_t = b_t$, which only involves matrix-vector multiplication, BLCG involves BLAS 3 operations and can better take advantage of parallelized BLAS implementations.

Algorithm 4.4 Block Lanczos CG

input: a matrix $A \in \mathbb{S}^m_{++}$, RHS $B \in \mathbb{R}^{n\times t}_{\neq 0}$ with rank $B = t \ll n$, error tolerance $\epsilon > 0$
initialize: $X_0 = 0$, $R_{-1} = [I_t, 0_{n-t}]^T$, $R_0 = B$, $k = -1$
1: **repeat**
2: $k \leftarrow k + 1$
3: $S_{k-1} = \left(R^T_{k-2}R_{k-2}\right)^{-1} R^T_{k-1}R_{k-1}$
4: $P_k = R_{k-1} + P_{k-1}S_{k-1}$
5: $T_k = \left(P^T_k AP_k\right)^{-1} R^T_{k-1}R_{k-1}$
6: $X_k = X_{k-1} + P_kT_k$MC
7: $R_k = R_{k-1} - AP_kT_k$
8: **until** $\max_{i=1,\ldots,t} \|r^{(k)}_i\| \leqslant \epsilon$
9: **return** X_k

4.4 Shifted systems and BLCG

Of particular relevance to the algorithms we later propose is the *shift-invariance property* of Krylov subspaces:

Lemma 4.2. *Krylov subspaces generated by symmetric matrices are* shift-invariant *in that, provided* $A = A^T$*, we have*

$$\mathcal{K}_m(A, r_0) = \mathcal{K}_m(A + \sigma I, r_0), \quad \forall \sigma \in \mathbb{R}. \tag{4.4.1}$$

Proof. We proceed via induction. Let $u \in \mathcal{K}_1(A + \sigma I, r_0)$. Then, $u = \beta_0 (A + \sigma I)^0 r_0 = \beta_0 r_0$ so $\mathcal{K}_1(A, r_0) = \mathcal{K}_1(A + \sigma I, r_0)$. Now, suppose $\mathcal{K}_{m-1}(A, r_0) = \mathcal{K}_{m-1}(A + \sigma I, r_0)$. Let $v \in \mathcal{K}_{m-1}(A +$

$\sigma I, r_0)$ and decompose the vector $u \in \mathcal{K}_m(A + \sigma I, r_0)$ as

$$u = \beta_m(A + \sigma I)v + \sum_{k=1}^{m-1} \beta_k A^{k-1} r_0 \tag{4.4.2}$$

$$= \beta_m Av + \sum_{k=1}^{m-1} \tilde{\beta}_k A^{k-1} r_0 \tag{4.4.3}$$

Thus, $u \in \mathcal{K}_m(A, r_0)$ and $\mathcal{K}_m(A, r_0) = \mathcal{K}_m(A + \sigma I, r_0)$. □

With the shift invariance property in mind, we turn our attention to the simultaneous solution of *shifted linear systems* via Lanczos Conjugate Gradients. These results are due to [6] and were initially motivated by Tikhonov regularization.

Definition 4.3. Given $A \in \mathbb{S}^n_{++}$ and nonzero vector b define the *seed system*

$$Ax = b \tag{4.4.4}$$

and the corresponding *family of shifted linear systems*

$$\mathcal{F}_{A,b} = \{(A + \sigma I)x_\sigma = b : \sigma \text{ such that } \kappa(A + \sigma I) < \kappa(A)\} \tag{4.4.5}$$

and denote a particular shifted system

$$A_\sigma x_\sigma = (A + \sigma I)x_\sigma = b. \tag{4.4.6}$$

Because of the shift invariance property, if we use the Lanczos procedure to construct an orthonormal basis $r_1, \ldots, r_m$ for $\mathcal{K}_m := \mathcal{K}_m(A, r_0)$, it will also span $\mathcal{K}^\sigma_m := \mathcal{K}_m(A_\sigma, r_0)$. Remarkably, having performed LCG on the seed system $Ax = b$, we can update the LCG coefficents in Algorithm 4.3 using only scalar operations. Having stored intermediate quantities, we are thus able to compute CG iterates for the shifted system using only vector operations. This allows for the rapid solution of all members of $\mathcal{F}_{A,b}$.

Birk and Frommer [2] extend this method to accommodate multiple right hand sides. In

this case, the *block seed system* is of the form $AX = B$, and we seek to solve all shifted systems

$$\mathcal{F}_{A,B} = \{(A + \sigma I)X_\sigma = B : \sigma \text{ such that } \kappa(A + \sigma I) < \kappa(A)\}.$$

Thus, having having employed the block Lanczos procedure to construct bases for $\mathcal{K}(A, B)$, we can then compute CG approximate solutions X_σ such that $A_\sigma X_\sigma = B$ for any value of σ using $O(t)$ operations, where t is the number of columns of B.

We present the following implementations of this procedure, which we term block Lanczos conjugate gradients (BLCG): Appendix C.4 uses the block Lanczos procedure to construct an orthonormal basis for for $\mathcal{K}(A, B)$. Appendix C.5 solves $(A + \sigma I)X_\sigma = B \in \mathbb{R}^{n\times t}$ for any σ in $O(nt)$ operations using the output of BLCG-seed. Figures 4.2 and 4.3 demonstrate timings for solving the seed system and shifted systems of the form:

$$\text{seed: } (\frac{1}{m}ZZ^T + \sigma_0 I_n)x = b, \quad \text{updates: } (\frac{1}{m}ZZ^T + \sigma_j I_n)x = b,\ \sigma_j > \sigma_0$$

with $\sigma_0 = .01$, for varying n. Each "LCG-update (Mx)" computation represents the solution of M additional shifted systems after completing BLCG-seed, with each update costing $O(n)$ operations.

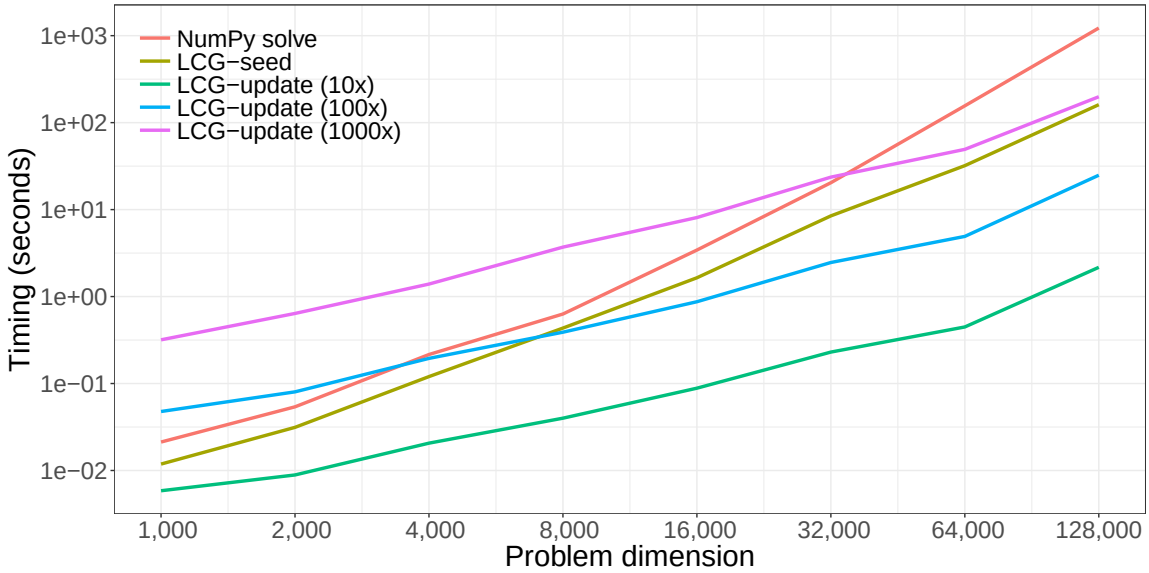


Figure 4.2: Performance of BLCG-seed and BLCG-update (log scale)

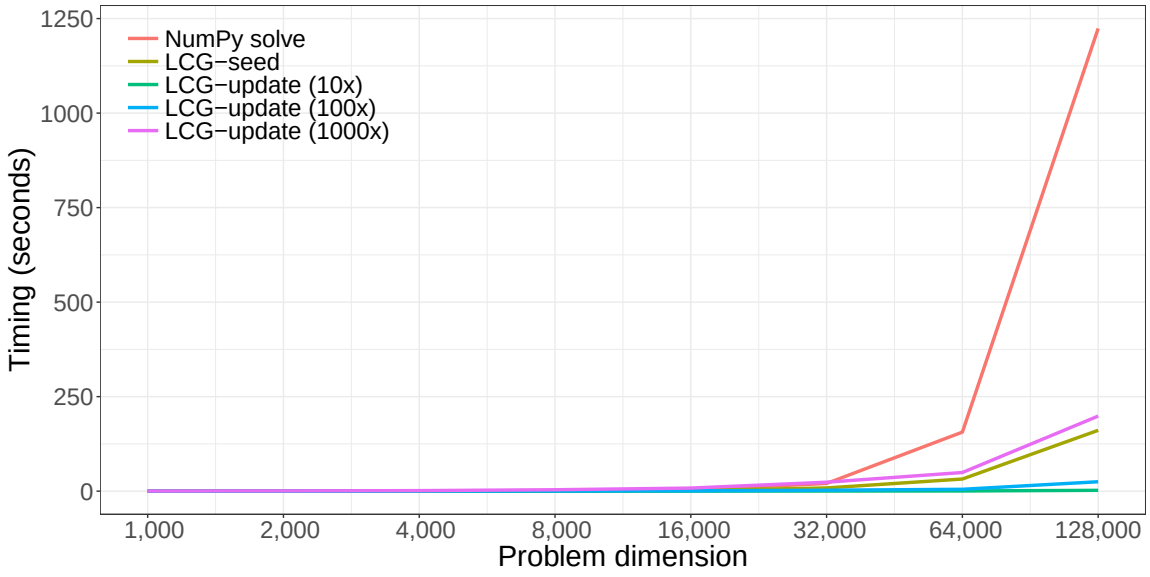


Figure 4.3: Performance of BLCG-seed and BLCG-update

Chapter 5

Approximating the log determinant

5.1 Overview

The algorithms we propose in Chapter 6, unlike those presented in Chapters 2-3, directly compute the log likelihood, rather than evaluating its gradients or computing BLUPs. We again consider the simple genetic variance components model

$$y \sim \mathcal{MVN}\left(\vec{0}, \gamma H\right), \quad H = \tau I + \frac{1}{m} Z Z^T,$$

where $\tau = \sigma_e^2/\sigma_g^2$ and $\gamma = \sigma_g^2$. The log-likelihood is then

$$\begin{aligned} \ell(\tau, \gamma | y) &\propto -\frac{1}{2}\left(\ln\det\{\gamma H\} + \gamma^{-1} y^T H^{-1} y\right) \\ &\propto \underbrace{n\log(\gamma^{-1})}_{(1)} - \underbrace{\ln\det\{H\}}_{(2)} - \underbrace{\gamma^{-1} y^T H^{-1} y}_{(3)}. \end{aligned}$$

We seek to compute the above likelihood efficiently. Term (1) is trivial and we can efficiently compute term (3) using CG followed by an $O(n)$ multiplication. The present section considers the evaluation of term (2), the log determinant. We propose to use Stochastic Lanczos quadrature (SLQ) [8] to approximate this quantity as described by [25]. The theory underlying SLQ has connections to the Spectral and Monte Carlo Newton Raphson methods described in Chapter 3 and the Krylov subspace theory developed in Chapter 4.

The method proceeds as follows. First, we rewrite $\ln\det\{H\} = \text{Tr } \log\{H\}$, the trace of the matrix logarithm of H, which is unique as $H \in \mathbb{S}^n_{++}$. We then apply a stochastic trace estimator of the form

$$\text{Tr } \{\log(H)\} \approx \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \check{v}_i^T \log(H) \check{v}_i = \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \check{v}_i^T Q \log(\Lambda) Q^T \check{v}_i$$

where $n_{\check{v}} \ll n$, $\check{v}_1, \ldots, \check{v}_{n_{\check{v}}}$ are randomly generated vectors, and H is written in terms of its spectral decomposition as $H = Q\log(\Lambda)Q^T$. For each $i = 1, \ldots, n_{\check{v}}$, we can recast the summand as the

Riemann-Stieltjes integral

$$\check{v}_i^T Q \log(\Lambda) Q^T \check{v}_i = \int_{\lambda_1}^{\lambda_n} \ln(t) d\mu(t)$$

with respect to the measure $\mu(t) = \sum_{j=1}^{\ell} \|e_j^T Q^T \check{v}_i\|^2 [\![\lambda_\ell \leq t]\!]$ where the eigenvalues of H are ordered from smallest to largest $\rho(H^{-1}) = \lambda_1 \leq \cdots \leq \lambda_m = \rho(H)$. Finally this integral is approximated via the Gaussian quadrature

$$\int_{\lambda_1}^{\lambda_n} \ln(t) d\mu(t) \approx \sum_{\ell=1}^{m} \omega_\ell \ln(x_\ell)$$

with nodes and weights obtained by applying the Lanczos process to construct a basis for $\mathcal{K}_m(H, \check{v}_i)$. Our presentation borrows heavily from that of [25], to which we refer the reader for further details. Here, we highlight key elements of the underlying theory.

5.2 Stochastic trace estimation

There exist a variety of Monte Carlo methods for efficient approximation of Tr H using only matrix-vector multiplications, the first of which is attributed to [12]. These methods, recognizing the trace can be written as the sum

$$\text{Tr } H = \sum_{i=1}^{n} e_i^T H e_i,$$

generate random *probing vectors* $\check{v}_1, \ldots, \check{v}_{n_{\check{v}}}$ with unit norm and zero expectation such that $\mathbb{E}[\check{v}_i^T H \check{v}_i] = \frac{1}{n_{\check{v}}} \text{Tr } H$ for each i. An unbiased stochastic estimator is thus given by

$$\text{Tr } H \approx \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \check{v}_i^T H \check{v}_i.$$

[1] and [22] provide bounds on several such estimators. Following [25], we generate probing vectors $\check{v}_i = \check{v}_i'/\|\check{v}_i'\|$, where each $\check{v}_i' \in \{-1, 1\}^n$ is a Rademacher random vector. As the matrix logarithm agrees with the scalar logarithm at the diagonal entries of $Q\Lambda Q^T = H \in \mathbb{S}_{++}^n$, an unbiased estimator is given by

$$\text{Tr } \log\{H\} \approx \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \check{v}_i^T Q \log(\Lambda) Q^T \check{v}_i,$$

where $\{\log(\Lambda)\}_{i,j} = \delta_{i,j} \ln \lambda_j$ for $\lambda_j \in \sigma(H)$. We later demonstrate via numerical experiment that a relatively small number ($n_{\check{v}} \approx 10 - 40$) of probing vectors is needed to achieve reasonable accuracy in the context of the problem of interest. Of course, computing the spectral decomposition of H takes $O(n^3)$ operations, and if we had access to $\sigma(H)$, we could compute Tr $\log\{H\}$ exactly in $O(n)$ operations. Remarkably, $\check{v}_i^T Q \log(\Lambda) Q^T \check{v}_i$ can be efficiently approximated with any such expensive operations via Gaussian quadrature, which we describe in the following section.

5.3 Lanczos polynomials and Lanczos Quadrature

Recall that, given $b \neq 0$, $A \in \mathbb{S}^n_{++}$, the Lanczos process yields unitary $U_m \in \mathbb{R}^{n \times m}$, tridiagonal $T_m = T_m^T \in \mathbb{R}^{m \times m}$ such that $AU_m = U_m T_m$ and col $U_m = \mathcal{K}_m(A, b)$. $m \ll n$ is chosen such that the corresponding m^{th} step CG approximation x_m is such that $\|Ax_m - b\| \leq \epsilon$ for some tolerance $\epsilon > 0$. Denoting the first column of U by $u_1 = b/\|b\|$, subsequent columns can be expressed in terms of a sequence of orthogonal polynomials in A as

$$u_k = p_{k-1}(A) u_1$$

for $k = 2, \ldots, m$. The k^{th} *Lanczos polynomial* p_k is the characteristic polynomial of the corresponding Jacobi matrix T_k consisting of the first k rows and columns of T_m. [8] demonstrate that these polynomials are orthogonal with respect to the spectral measure

$$\mu_b(t) = \sum_{j=1}^{\ell} \|e_j^T Q^T b\|^2 \left[\!\left[\lambda_\ell \leq t\right]\!\right], \quad \ell = 1, \ldots, n,$$

where $A = Q \Lambda Q^T$ is the spectral decomposition.

Thus, casting expressions of the form $v^T Q \log(\Lambda) Q^T v$ as Riemann-Stieltjes integrals

$$v^T Q \log(\Lambda) Q^T v = \int_{\lambda_1}^{\lambda_n} \ln(t) d\mu_v(t),$$

the Lanczos decomposition provides us with an efficient means of approximation via Gaussian quadrature. The roots of the m^{th} orthogonal polynomial are those of characteristic polynomial of the Jacobi matrix T_m. Thus, as in the Golub-Welsch algorithm, the eigendecomposition of

T_m can be used to compute the quadrature weights ω_ℓ and nodes x_ℓ, $\ell = 1, \ldots, m$ [10]. The eigendecomposition is easily computed as T_m is symmetric tridiagonal and $m \ll n$ is small.

5.4 Stochastic Lanczos Quadrature to approximate $\log\det A$

Stochastic Lanczos Quadrature (SLQ) consists of employing Lanczos quadrature to compute the summands of the stochastic trace estimator for spectral functions of matrices. Our estimator for $\log\det\{A\}$ will be

$$\begin{aligned}
\text{Tr } \{\log(H)\} &\approx \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \check{v}_i^T Q \log(\Lambda) Q^T \check{v}_i \\
&= \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \int_{\lambda_1}^{\lambda_n} \ln(t) d\mu_{\check{v}_i}(t) \\
&\approx \frac{n}{n_{\check{v}}} \sum_{i=1}^{n_{\check{v}}} \sum_{\ell=1}^{m} \omega_{i,\ell} \ln(\theta_{i,\ell}),
\end{aligned}$$

where $\theta_{i,\ell} = \{D_i\}_{\ell,\ell}$, $\omega_{i,\ell} = \{e_1^T P_i\}_\ell$ and $P_i D_i P_i^T = T_i$ is the spectral decomposition of the Jacobi matrix generated by applying the Lanczos decomposition to system $Ax_i = \check{v}_i$ for each random vector $\check{v}_i$. Collecting each random vector into the matrix $\check{V} = (\check{v}_1, \ldots, \check{v}_{n_{\check{v}}})$, we can simultaneously generate the Jacobi matrices $T_1, \ldots, T_{n_{\check{v}}}$ via our previously described BLCG-seed algorithm (Appendix C.4). We describe such a procedure, based on that of [25], in Section 5.1. Additionally, we implement a function to approximate Tr $f(A)$ which operates on the output of BLCG-seed applied to $AX = \check{V}$ in Appendix C.6.

Algorithm 5.1 Approximate log determinant via Stochastic Lanczos Quadrature (SLQ)

input: a matrix $A \in \mathbb{S}^n_{++}$, number of probing vectors $n_{\check{v}}$, error tolerance $\epsilon > 0$
initialize: $n \times n_{\check{v}}$ array $\check{V}$, $\Gamma = 0$
1: **for** $k = 1$ to $n_{\check{v}}$ **do**
2: generate Rademacher random n-vector $\check{v}$
3: $\check{V}_{.,k} \leftarrow \check{v}/\|\check{v}\|$
4: **end for**
5: compute Jacobi matrix T_k such that $AU_k = U_k T_k$ with col $U_k = \mathcal{K}_m(A, \check{V}_{.,k})$ for $k = 1, \ldots, n_{\check{v}}$ by applying `BLCG-seed` to $AX = \check{V}$ at tolerance ϵ
6: **for** $k = 1$ to K **do**
7: $d,\ P \leftarrow$ `eigen_symmetric_tridiagonal`(T_k)
8: $p \leftarrow [p^2 :\ p \in P_{1,.}]$
9: $\theta \leftarrow [\log(t) :\ t \in d]$
10: $\Gamma \leftarrow \Gamma + \theta^T p$
11: **end for**
12: **return** $\log\det A \approx \frac{n}{n_{\check{v}}} \cdot \Gamma$

In Figure 5.1 we present results from numerical experiments examining the performance of the SLQ trace estimator as a function of problem dimension, the number of random probing vectors, and the spectral conditioning of the covariance matrix. Experiments compared the true log determinant (computed using direct methods) to SLQ approximations for 100 Monte Carlo replicates of the matrix $H = \frac{1}{m} ZZ^T + \frac{1-\sigma_g^2}{\sigma_g^2} I$ (randomly simulating genotypes). The top panel corresponds to a moderate heritable variance component value of $\sigma_g^2 = .25$ whereas the bottom panel corresponds to a moderate heritable variance component value of $\sigma_g^2 = .40$. Results suggest that, in addition to decreasing with the number of probing vectors, error decreases with problem dimension. Additionally, error appears to depend on the conditioning of the covariance matrix such that accuracy decreases as the heritable variance component increases.

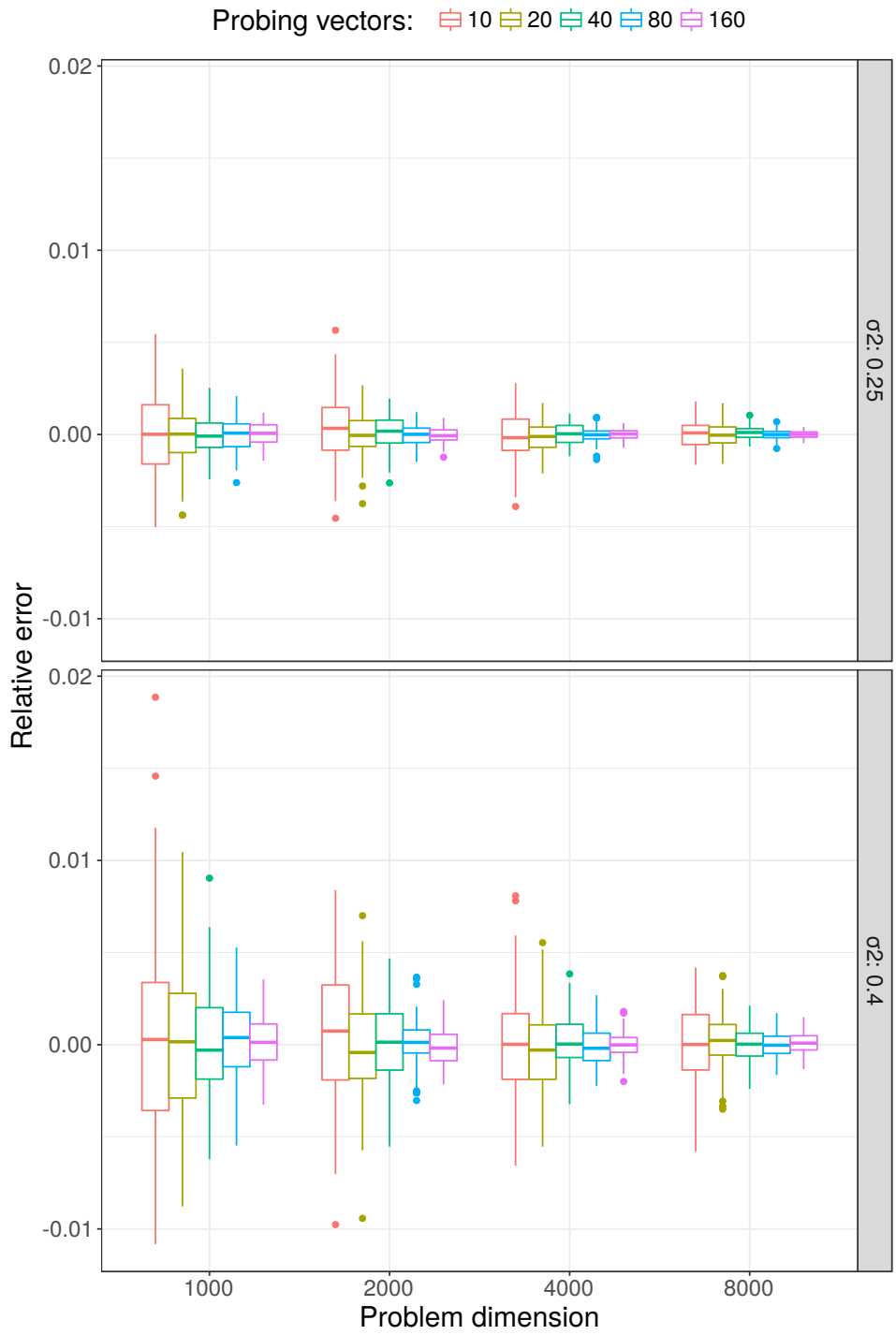


Figure 5.1: SLQ log determinant approximation accuracy

Chapter 6

Proposed algorithm

6.1 Putting everything together

Consider again the simple genetic variance components model

$$y \sim \mathcal{MVN}\left(\vec{0}, \gamma H\right), \quad H = \tau I_n + \frac{1}{m} Z Z^T, \tag{6.1.1}$$

where $\tau = \sigma_e^2/\sigma_g^2$, $\gamma = \sigma_g^2$, and the log-likelihood is

$$\ell(\tau, \gamma | y) \propto n \log(\gamma^{-1}) - \ln \det\{H\} - \gamma^{-1} y^T H^{-1} y.$$

We have established efficient methods for approximating $\ln \det\{H\}$ and $\gamma^{-1} y^T H^{-1} y$, both of which would be prohibitively expensive for large n using direct methods. Here we demonstrate that these methods are particularly well-suited to estimation of the model (6.1.1) due to the shift invariance property of Krylov subspaces. In particular, the parameter space Θ over which we seek to optimize (6.1.1) defines a family of shifted linear systems. Thus, after evaluating the likelihood at some point (τ, γ), we can rapidly evaluate the likelihood for other $(\tau', \gamma') \in \Theta$. In what follows, we propose a novel algorithm for the genetics variance components problem based on this observation and present the results of numerical experiments. Additionally, we suggest alternative procedures that would also take advantage of this shift-invariance property. Finally, we discuss the limitations of our proposed methods and suggest directions for future work.

6.2 A Lanczos method for genomics variance components estimation

The covariance γH in (6.1.1) comprises a convex combination of the relatedness matrix $\frac{1}{m} Z Z^T$ and the identity. As a consequence of these constraints (i.e., $\sigma_e^2 + \sigma_g^2 = 1$, $\sigma_e^2, \sigma_g^2 > 0$) we

have that

$$\gamma\,(\tau+1)=1 \quad \Longrightarrow \quad \gamma=\frac{1}{\tau+1}.$$

We can thus represent the likelihood as a function of a single parameter:

$$\ell(\tau|y) \propto n\log(\tau+1)-\ln\det\{H(\tau)\}-(\tau+1)y^T H^{-1}(\tau)y, \tag{6.2.1}$$

where the notation $H(\tau)$ is intended to emphasize the dependence of the covariance on the unknown variance component: $H(\tau)=\tau I_n+\frac{1}{m}ZZ^T$. Suppose we've computed $\ell(\tau_0|y)$ for some $\tau_0\in\Theta:=\left\{\tau:\frac{1}{\tau+1},\frac{\tau}{\tau+1}>0\right\}$ by using Lanczos Conjugate Gradients to compute $H^{-1}(\tau_0)y$ and Stochastic Lanczos Quadrature to approximate $\ln\det\{H(\tau_0)\}$ with probing vectors $\check{v}_1,\ldots,\check{v}_K$. We thus have constructed orthonormal bases $U_0,\ldots,U_K$ with corresponding Jacobi matrices $T_0,\ldots,T_k$ for the block Krylov subspace $\mathcal{K}(H(\tau_0),B)$, where $B=(y,\check{v}_1,\ldots,\check{v}_K)$. Evaluating the likelihood at an alternative value $\tau'=\tau_0+\sigma$,

$$\ell(\tau'|y) \propto n\log(\tau')-\ln\det\{H(\tau_0)+\sigma I\}-(\tau')y^T\left(H(\tau_0)+\sigma I\right)^{-1}y,$$

amounts to applying the Lanczos methods to the shifted system $H(\tau_0)+\sigma I$. As a consequence of the shift invariance property, we can reuse the bases $U_0,\ldots,U_K$. The updated Jacobi matrices are simply $T_i\leftarrow T_i+\sigma I$.

For an observed phenotype y, let $s^2_{\max}$ indicate the maximal value of σ_g^2 that we would consider plausible based on previous knowledge. Reparameterizing as $\tau_0=(1-s^2_{\max})/s^2_{\max}$, our effective parameter space becomes $\Theta'=\left\{\tau\geq\tau_0:\frac{1}{\tau+1},\frac{\tau}{\tau+1}>0\right\}$[1]. Together with $B=(y,\check{v}_1,\ldots,\check{v}_{n_{\check{v}}})$, this defines $n_{\check{v}}+1$ families of shifted linear systems

$$\mathcal{F}_{H(\tau_0),B_{\cdot,j}}=\left\{(H(\tau_0)+\sigma I)X_\sigma=B_{\cdot,j}:\sigma \text{ such that } (\tau_0+\sigma)\in\Theta'\right\},\quad j=0,\ldots,K.$$

After computing $\ell(\tau_0|y)$, we can compute $\ell(\tau|y)$ for any $\tau\in\Theta'$ using vector operations as in

[1]We choose $\tau_0=(1-s^2_{\max})/s^2_{\max}$ such that all shifted covariance matrices will be of the form $H_\sigma=H(\tau_0)+\sigma I$ for positive σ. As a result, we then have $\kappa(H_\sigma)<\kappa(H(\tau_0))$. Denoting CG approximate solutions to the seed system and a shifted system $x_0\approx H(\tau_0)^{-1}y$, $x_\sigma\approx H_\sigma^{-1}y$, we then expect that that $\left\|H_\sigma^{-1}y-x_\sigma\right\|<\left\|H(\tau_0)^{-1}y-x_0\right\|$. That is, we start with the value $\tau_0\in\Theta'$ that results in the most ill-conditioned linear system among those under consideration to ensure convergence of the Lanczos process when applied to the shifted systems.

Appendix C.5. We thus arrive at the following procedure, consisting of two steps. The first step consists of using the block Lanczos process to generate orthonormal bases/Jacobi matrices:

1. Given a range $\Theta = [s^2_{\min}, s^2_{\max}]$ of possible values for the heritable variance component σ^2_g, GRM $\frac{1}{m}ZZ^T$, and observed phenotype vector y, set $\tau_0 = (1 - s^2_{\max})/s^2_{\max}$ and define $H_0 = \frac{1}{m}ZZ^T + \tau_0 I$

2. Generate $n_{\check{v}}$ normed Rademacher random vectors $\check{v}_1, \dots \check{v}_{n_{\check{v}}}$

3. Use BLCG-seed (Appendix C.4) to compute Lanczos decompositions $AU_k = U_k T_k$ for the $n_{\check{v}} + 1$ seed systems
$$H_0 x_0 = y,\ H_0 x_1 = \check{v}_1,\ \dots\ , H_0 x_k = \check{v}_{n_{\check{v}}}$$

4. For $k = 1, \dots, n_{\check{v}}$, compute the eigendecomposition $T_k = P_k \Lambda_k P_k^{-1}$

Second, we perform a bisection-type maximization procedure on $\ell(\tau|y)$ for $\tau \in \Theta' = \left\{(1 - s^2)/s^2 \in \Theta\right\}$, evaluating the likelihood as follows:

1. Defining $\sigma = \tau - \tau_0$, update $T_0^\sigma = T_0 + \sigma I$ and, for $k > 0$, update $\Lambda_k^\sigma = \Lambda_k + \sigma I$

2. Use U_0, T_0^σ to compute the CG approximate solution x_σ to $H_\sigma x_\sigma = y$ as in (Appendix C.5)

3. Estimate the log determinant via SLQ as
$$\log\left(\det(A)\right) \approx \Gamma_\sigma = \frac{n}{n_{\check{v}}} \sum_{k=1}^{n_{\check{v}}} \left(\left[\log \lambda : \lambda \in \operatorname{diag}\Lambda\right]^T \left(e_1^T P_k\right) \circ \left(e_1^T P_k\right) \right)$$

4. Compute objective function as
$$\ell(\tau|y) = n \log(\tau + 1) - \Gamma_\sigma - (\tau + 1) y^T x_\sigma$$

We provide pseudo-code for this procedure, which we term Stochastic Block Lanczos Likelihood Estimation (SBLLE), in Appendix 6.1. A Python 3 implementation is provided in Appendix C.7.

Algorithm 6.1 Stochastic Block Lanczos Likelihood Estimation (SBLLE)

input: genomic relatedness matrix $K \in \mathbb{S}_+^m$,
phenotype $y \in \mathbb{R}^n$,
bounds $s^2_{\min}, s^2_{\max} \in (0,1)$ of search interval for σ_g^2,
number of probing vectors $n_{\check{v}}$,
number of subintervals k for line search

initialize: $\ell = \texttt{zeros}(k), \hat{\tau} = 0, \tau_{\text{upper}} = +\infty, \tau_{\text{lower}} = -\infty$

$S^2 = \left\{(s^2_{\max} - j \cdot (s^2_{\max} - s^2_{\min})/(k-1) : \text{ for } j = 0, \ldots, k-1\right\}$
$\check{V}' = \left\{\texttt{rademacher}(n) \text{ for } j = 1, \ldots, n_{\check{v}}\right\}$
$\check{V} = \left\{\check{v}'/\|\check{v}'\| \text{ for } \check{v}' \in \check{V}'\right\}$
$\tau_0 = (1 - s^2_{\max})/s^2_{\max}$
$\{(T_i, U_i)\}_{i=0}^{n_{\check{v}}} = \texttt{BLCG\_seed}\ (K + \tau_0 I_n, (y, \check{v}_1, \ldots, \check{v}_{n_{\check{v}}}))$
$\Gamma \leftarrow 0$
for $i = 1, \ldots, n_{\check{v}}$ **do**
  $\Lambda_i, P_i \leftarrow \texttt{eigen\_symmetric\_tridiagonal}(T_i)$
end for
while $|\tau_{\text{upper}} - \tau_{\text{lower}}| > \varepsilon$ **do**
  $\Gamma \leftarrow 0$
  $\Sigma \leftarrow \left\{(1 - s^2)/s^2 \text{ for } s^2 \in S^2 : s^2\right\}$
  for $i = 0, \ldots, k-1$ **do**
    $y^T H y \leftarrow \texttt{BLCG\_update}(T_0, U_0, \Sigma[i])$
    for $j = 1, \ldots, n_{\check{v}}$ **do**
      $p \leftarrow [p^2 : \ p \in P_i e_1]$
      $w \leftarrow \{\log(\lambda) : \ \lambda \in \Lambda_j + \Sigma[j] I\}$
      $\Gamma \leftarrow \Gamma + w^T p$
    end for
    $\ell[i] \leftarrow n \ln(\tau_0 + \Sigma[i] + 1) - \frac{n}{n_{\check{v}}} \Gamma - (\tau_0 + \Sigma[i] + 1) y^T H y$
  end for
  $j_{\text{opt}} \leftarrow \arg\max_j(\ell[j])$
  $j_{\text{upper}} \leftarrow \arg\max_{j:\ell[j] < \ell[j_{\text{opt}}]} \{\ell[i]\}$
  $j_{\text{lower}} \leftarrow \arg\max_{j:\ell[j] < \ell[j_{\text{upper}}]} \{\ell[i]\}$
  $\tau_{\text{lower}} \leftarrow \tau_0 + \Sigma[j_{\text{lower}}]$
  $\tau_{\text{upper}} \leftarrow \tau_0 + \Sigma[j_{\text{upper}}]$
  $\hat{\tau} \leftarrow \tau_0 + \Sigma[j_{\text{opt}}]$
end while
return $\hat{\tau}$

6.3 Performance and numerical experiments

There are a number of desirable properties of our proposed algorithm. Assuming $n_{\check{v}}$ probing vectors, block Lanczos applied to the block system $HX = B = (y, \check{v}_1, \ldots, \check{v}_{n_{\check{v}}})$ requires the $O(n^2 n_{\check{v}})$ operations per iteration (assuming the condition number of the GRM is fixed). As the termination criterion for the Lanczos process is identical to that of classical Conjugate Gradients, the number of required iterations to achieve accuracy at a specified tolerance is a function of the spectral condition

number of A [3]. In our experiments, it typically required ≈ 20 iterations of the Lanczos process to achieve accuracy to the ninth decimal place (when solving linear systems). Additionally, unlike Newton-methods, choice of starting values does not present a challenge as, after the initial Lanczos step, we can cheaply perform a grid search over the entire parameter space.

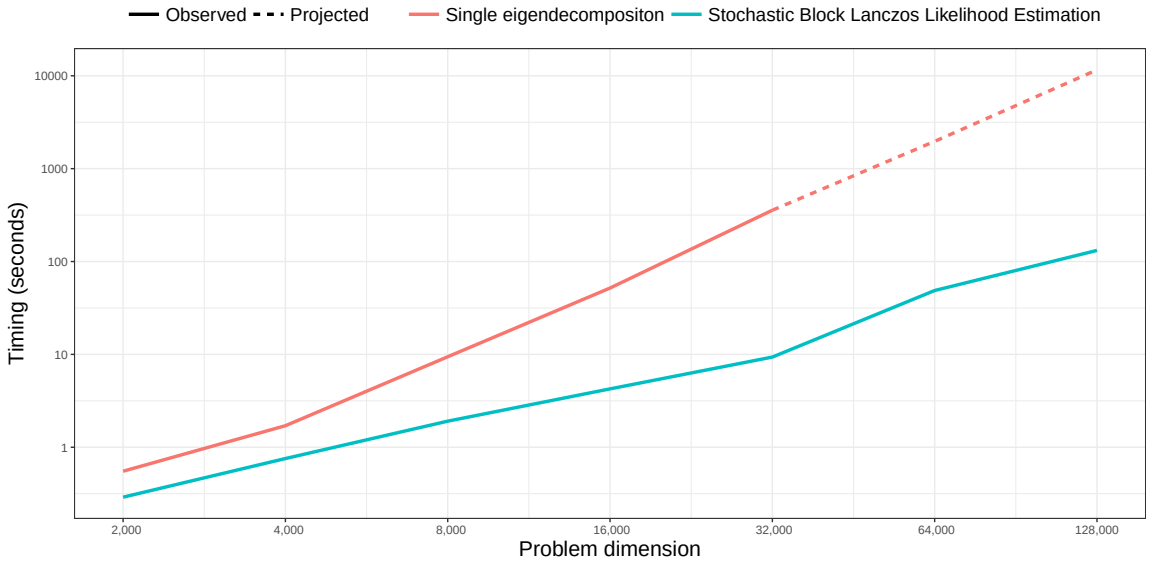


Figure 6.1: Timings of complete SBLLE versus single eigendecomposition (log scale)

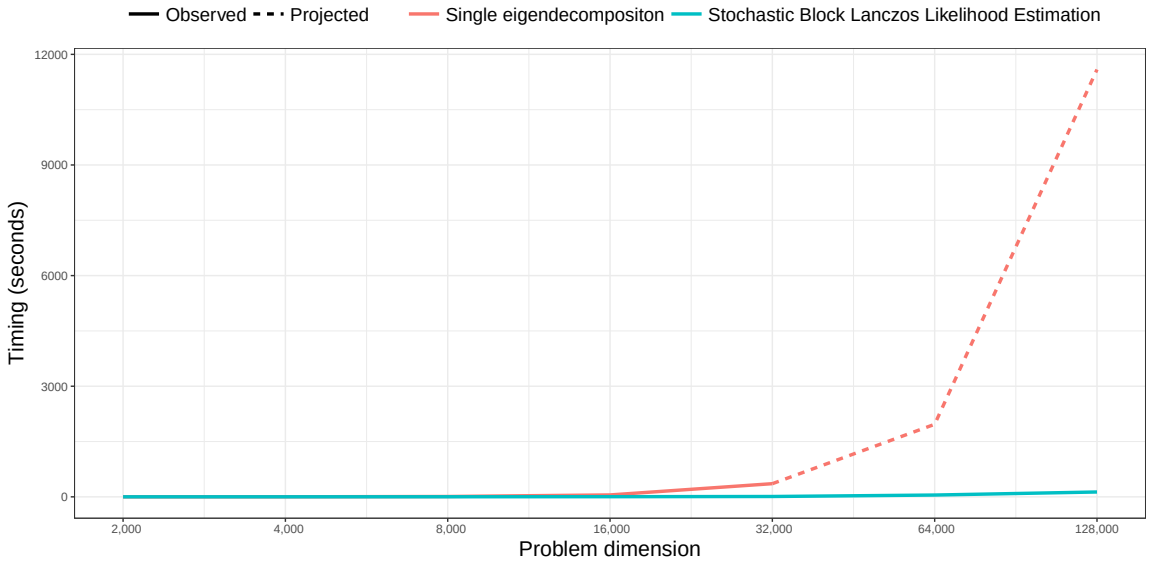


Figure 6.2: Timings of complete SBLLE versus single eigendecomposition

Figures 6.1-6.2 compare timings for a single eigendecomposition (as is required for the

spectral NR algorithm of Section 3.2) to the complete execution of the SBLLE algorithm using 40 probing vectors and enforcing precision to the fifth decimal place (with respect to the variance component estimates). These experiments were conducted on an Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz with 32 physical cores and 1 terabyte of RAM, assuming a maximal heritable variance component value of 0.30. The code we used to generate data is presented in Appendix C.8.

Figure 6.3 compares the estimates of three stochastic likelihood estimation procedures to those obtained from standard Newton Raphson (as implemented in Appendix C.2). Monte Carlo Newton Raphson (MCNR) is as described in Section 3.3 and as implemented in Appendix C.3. SBLLE is as described in Section 6.2 and as implemented in Appendix C.7. SBLLE-rand differs from SBLLE only in the grid search method: uniformly distributed nodes, rather than equispaced nodes, are considered in each L-section iteration (including the endpoints in each case). Experiments were conducted using 1000 simulated data sets, each consisting of 8,000 individuals and 9,600 markers, considering a maximal heritable variance component value of $s^2_{\max} = .25$. The total number of probing vectors was set to 30 for all methods. Results suggest little difference between the two proposed Lanczos methods but increased accuracy of the Lanczos methods compared to the MCNR method.

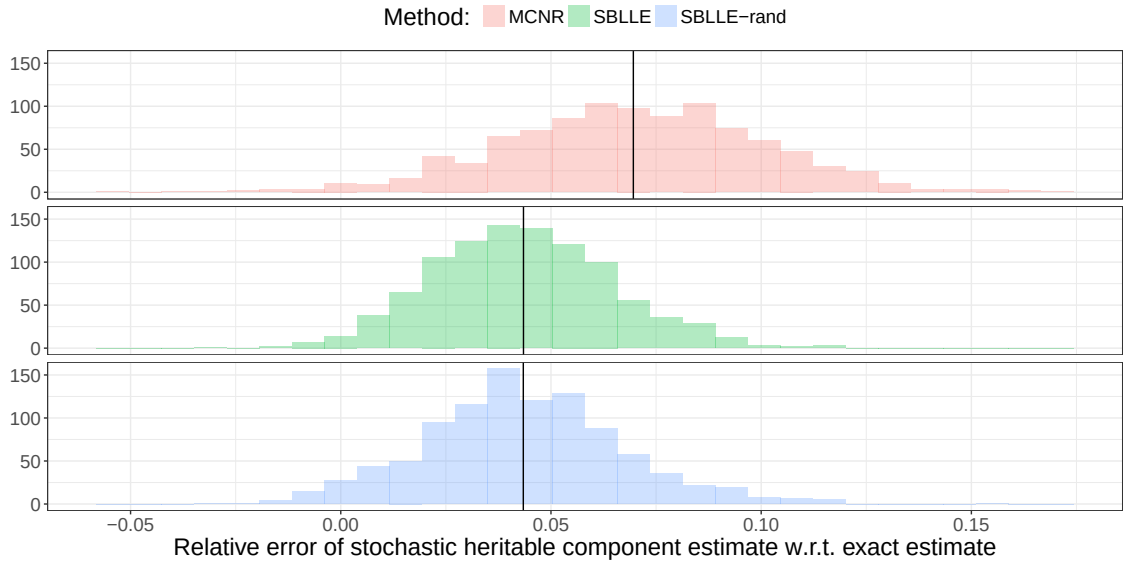


Figure 6.3: Accuracy of stochastic algorithms with respect to exact algorithms

Chapter 7

Conclusion

7.1 Summary

The present thesis examined algorithms for likelihood estimation of the simplified genomic variance components model

$$\begin{aligned}(y|u) &\sim \mathcal{MVN}(m^{-1/2}Zu, \sigma_e^2 I_n),\\ u_i &\overset{iid}{\sim} \mathcal{N}(0, \sigma_g^2), \quad i = 1, \ldots, m,\end{aligned}$$

where y is a measured phenotype, Z is a matrix of measured genotypes, u is a vector of latent genetic effects, and we seek to estimate the variance components $\theta = (\sigma_g^2, \sigma_e^2)^T$. Though the parameter space $\Theta = (0, 1)^2$ is low dimensional, the marginal covariance matrix $\Sigma_\theta = \sigma_g^2 m^{-1} ZZ^T + \sigma_e^2 I_n$ is large, dense, and unstructured. As a result, likelihood estimation of the variance components poses a challenging computational problem, with the order of the covariance matrix Σ_θ already surpassing $n = 1,000,000$ in data sets of interest [14].

After introducing the problem, we discussed two previous approaches to efficient genomic variance component estimation. The Spectral Newton Raphson (Spectral NR) algorithm (Section 3.2, Appendix C.2) uses clever algebraic manipulations to avoid all but one $O(n^3)$ computation. However, for large n, even a single $O(n^3)$ operation becomes prohibitively expensive (see Figures 6.1-6.2). The Monte Carlo Newton Raphson (MCNR) algorithm (Section 3.3, Appendix C.3) avoids all $O(n^3)$ operations by using a stochastic procedure to implicitly approximate the trace of matrix function and, as implemented by [17], by using the method of Conjugate Gradients to solve linear systems. Our proposed method takes a similar approach, but takes advantage of additional structure in the problem formulation and recent advances in trace approximation.

Chapter 4 introduced the Lanczos process, emphasizing its relationship to the method of Conjugate Gradients and how the shift-invariance property of Krylov subspaces allows for the si-

multaneous solution of shifted families of linear systems. Chapter 5 introduced stochastic trace estimation and stochastic Lanczos quadrature for approximating the log determinant of the covariance matrix. Here, we emphasized how the Lanczos method of the previous chapter could be used to efficiently compute nodes and weights for the Gaussian quadratures approximating each of the summands of the trace estimator.

Finally, in Chapter 6, we proposed the Stochastic Block Lanczos Likelihood Estimation (SBLLE) procedure. In contrast to Newton-type methods, our method repeatedly evaluates the log likelihood function (rather than its derivatives). This is accomplished by using the Lanczos process to obtain approximations to the log determinant and linear system terms of the likelihood objective

$$\ell(\theta|y) \propto -\left(\ln\det\{\Sigma_\theta\} + y^T\Sigma_\theta^{-1}y\right).$$

By suitably reparameterizing the variance components, we are able to identify the parameter space of interest with families of shifted linear systems, and, again exploiting shift-invariance, are then able to reevaluate the likelihood for new values of the variance components using vector operations.

In this respect, our method combines ideas present in both the Spectral NR and MCNR algorithms. As in Spectral NR, we perform a single set of expensive computations, the output of which we then use to repeatedly evaluate all remaining necessary terms until convergence is achieved. As in MCNR, we substitute stochastic and iterative procedures for direct matrix operations, reducing the total computational cost to $O(n^2\eta)$, where η depends on $\tilde{\kappa} = \max_{\theta\in\Theta}\kappa(\Sigma_\theta)$. However, in contrast to MCNR, we avoid the additional $O(n^2\eta)$ computations involved in computing the BLUPs of the latent variables at each iteration (see Section 3.3), and, by taking advantage of shift-invariance, require only a single $O(n^2\eta)$ step. Numerical experiments demonstrated that this increased computational efficiency does not come at the cost of decreased accuracy (Figure 6.3).

7.2 Limitations and future directions

There are three primary limitations to the present thesis, each of which presents future directions for research. First, our proposed method does not address the pertinent issue of the high memory cost ($O(n^2)$) involved in working with large relatedness matrices. As a result, expensive

compute nodes with multiple terabytes of random access memory are necessary to work with large samples, with intractable amounts of memory required to work with the largest data sets. This shortcoming is shared by all of the commonly used software implementations of the algorithms we reviewed. Extending our proposed method to operate in parallel across multiple nodes would help mitigate the need for single high-memory nodes. Second, our proposed method does not allow for covariates, in which case unbiased estimators for the variance components are obtained by maximizing the residual maximum likelihood (REML) criterion. This involves, for r linearly independent covariates $X = (x_1, \ldots, x_r)$, optimizing the likelihood corresponding to the random vector $K^T y$ with marginal distribution

$$K^T y \sim \mathcal{MVN}\left(\vec{0}, K^T \Sigma_\theta K\right),$$

where K^T projects to the $(n-r)$-dimensional orthogonal complement of X in $\mathbb{R}^n$. We are currently working on extending SBLLE to the REML case. Finally, the implementations presented in Appendix C serve only as proof-of-concept and would be difficult to employ in the context of primary research in genetics. As such, we aim to incorporate our proposed methods into existing software packages that have mature routines for handling and analyzing genomic data.

Bibliography

[1] Haim Avron and Sivan Toledo. "Randomized algorithms for estimating the trace of an implicit symmetric positive semi-definite matrix". en. In: *Journal of the ACM* 58.2 (Apr. 2011), pp. 1–34. ISSN: 00045411. DOI: 10.1145/1944345.1944349.

[2] Sebastian Birk and Andreas Frommer. "A deflated conjugate gradient method for multiple right hand sides and multiple shifts". en. In: *Numerical Algorithms* 67.3 (Nov. 2014), pp. 507–529. ISSN: 1017-1398, 1572-9265. DOI: 10.1007/s11075-013-9805-9.

[3] Ake Björck. *Numerical methods in matrix computations*. Vol. 59. Springer, 2015.

[4] Douglas Scott Falconer. *Introduction to quantitative genetics*. Oliver and Boyd; Edinburgh; London, 1960.

[5] R. A. Fisher. "The Correlation between Relatives on the Supposition of Mendelian Inheritance." en. In: *Earth and Environmental Science Transactions of The Royal Society of Edinburgh* 52.2 (1919), pp. 399–433. ISSN: 2053-5945, 0080-4568. DOI: 10.1017/S0080456800012163.

[6] Andreas Frommer and Peter Maass. "Fast CG-Based Methods for Tikhonov-Phillips Regularization". en. In: (1997), p. 22.

[7] A. L. García-Cortés et al. "Variance component estimation by resampling". en. In: *Journal of Animal Breeding and Genetics* 109.1-6 (Jan. 1992), pp. 358–363. ISSN: 09312668. DOI: 10.1111/j.1439-0388.1992.tb00415.x.

[8] Gene H. Golub and Gérard Meurant. *Matrices, Moments and Quadrature with Applications:* en. Princeton: Princeton University Press, Jan. 2009. ISBN: 978-1-4008-3388-7. DOI: 10.1515/9781400833887.

[9] Gene H. Golub and Charles F. Van Loan. *Matrix computations*. Vol. 3. JHU Press, 2012.

[10] Gene H. Golub and John H. Welsch. “Calculation of Gauss quadrature rules”. en-US. In: *Mathematics of Computation* 23.106 (1969), pp. 221–230. ISSN: 0025-5718, 1088-6842. DOI: 10.1090/S0025-5718-69-99647-1.

[11] F. N. Gumedze and T. T. Dunne. “Parameter estimation and inference in the linear mixed model”. In: *Linear Algebra and its Applications* 435.8 (Oct. 2011), pp. 1920–1944. ISSN: 0024-3795. DOI: 10.1016/j.laa.2011.04.015.

[12] M. F. Hutchinson. “A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines”. In: *Communications in Statistics - Simulation and Computation* 19.2 (Jan. 1990), pp. 433–450. ISSN: 0361-0918. DOI: 10.1080/03610919008812866.

[13] Cornelius Lanczos. “Solution of systems of linear equations by minimized iterations”. In: *J. Res. Nat. Bur. Standards* 49.1 (1952), pp. 33–53.

[14] James J. Lee et al. “Gene discovery and polygenic prediction from a genome-wide association study of educational attainment in 1.1 million individuals”. En. In: *Nature Genetics* 50.8 (Aug. 2018), p. 1112. ISSN: 1546-1718. DOI: 10.1038/s41588-018-0147-3.

[15] Christoph Lippert et al. “FaST linear mixed models for genome-wide association studies”. en. In: *Nature Methods* 8.10 (Oct. 2011), pp. 833–835. ISSN: 1548-7105. DOI: 10.1038/nmeth.1681.

[16] Po-Ru Loh et al. “Efficient Bayesian mixed-model analysis increases association power in large cohorts”. en. In: *Nature Genetics* 47.3 (Mar. 2015), pp. 284–290. ISSN: 1546-1718. DOI: 10.1038/ng.3190.

[17] Po-Ru Loh et al. “Efficient Bayesian mixed-model analysis increases association power in large cohorts”. In: *Nature Genetics* 47 (Feb. 2015), p. 284.

[18] Po-Ru Loh et al. “Mixed-model association for biobank-scale datasets”. In: *Nature genetics* (2018), p. 1.

[19] Teri A. Manolio et al. “Finding the missing heritability of complex diseases”. en. In: *Nature* 461.7265 (Oct. 2009), pp. 747–753. ISSN: 1476-4687. DOI: 10.1038/nature08494.

[20] Charles McCulloch. *Generalized, linear, and mixed models*. Hoboken, New Jersey: John Wiley & Sons, 2008. ISBN: 978-0-470-07371-1.

[21] Tinca J. C. Polderman et al. “Meta-analysis of the heritability of human traits based on fifty years of twin studies”. en. In: *Nature Genetics* 47.7 (July 2015), pp. 702–709. ISSN: 1546-1718. DOI: 10.1038/ng.3285.

[22] Farbod Roosta-Khorasani and Uri Ascher. “Improved Bounds on Sample Size for Implicit Matrix Trace Estimators”. en. In: *Foundations of Computational Mathematics* 15.5 (Oct. 2015), pp. 1187–1212. ISSN: 1615-3383. DOI: 10.1007/s10208-014-9220-1.

[23] Shayle R Searle, George Casella, and Charles E McCulloch. *Variance components*. Vol. 391. Citation Key: searle2009variance. John Wiley & Sons, 2009.

[24] Cathie Sudlow et al. “UK Biobank: An Open Access Resource for Identifying the Causes of a Wide Range of Complex Diseases of Middle and Old Age”. en. In: *PLOS Medicine* 12.3 (Mar. 2015), e1001779. ISSN: 1549-1676. DOI: 10.1371/journal.pmed.1001779.

[25] Shashanka Ubaru, Jie Chen, and Yousef Saad. “Fast Estimation of $tr(f(A))$ via Stochastic Lanczos Quadrature”. en. In: *SIAM Journal on Matrix Analysis and Applications* 38.4 (Jan. 2017), pp. 1075–1099. ISSN: 0895-4798, 1095-7162. DOI: 10.1137/16M1104974.

[26] Peter M. Visscher, William G. Hill, and Naomi R. Wray. “Heritability in the genomics era–concepts and misconceptions”. eng. In: *Nature Reviews. Genetics* 9.4 (Apr. 2008), pp. 255–266. ISSN: 1471-0064. DOI: 10.1038/nrg2322.

[27] Jian Yang et al. “Advantages and pitfalls in the application of mixed model association methods”. In: *Nature genetics* 46.2 (Feb. 2014), pp. 100–106. ISSN: 1061-4036. DOI: 10.1038/ng.2876.

[28] Jian Yang et al. “Common SNPs explain a large proportion of the heritability for human height”. In: *Nature genetics* 42.7 (2010), p. 565.

[29] Xiang Zhou and Matthew Stephens. “Genome-wide Efficient Mixed Model Analysis for Association Studies”. In: *Nature genetics* 44.7 (June 2012), pp. 821–824. ISSN: 1061-4036. DOI: 10.1038/ng.2310.

[30] Shengxin Zhu, Tongxiang Gu, and Xingping Liu. “Information Matrix Splitting”. In: *arXiv:1605.07646 [math, stat]* (May 2016). arXiv: 1605.07646.

Appendix A

Notation and glossary

Throughout this thesis, we assume we are working with real numbers unless explicitly stated otherwise.

A.1 Symbols

$\hat{\theta}$	estimated valued of parameter theta
$\tilde{u}$	BLUP of latent random variable u
$\check{v}$	computer generated (pseudo-)random vector
$\propto$	"depends on only ..." (i.e., $f(x) \propto z$ implies that $f(x)$ is an order-preserving affine function of z, and in the context of optimization can be regarded as "proportional to")
$A_{i,\cdot}$, $A_{\cdot,j}$	vectors comprising the i^{th} row and j^{th} column of a matrix A, respectively.
$[\![p]\!]$	the Iverson bracket, mapping a proposition p to $\{0, 1\}$ and equal to one iff p is true
δ_{ij}	the Kronecker delta: $\delta_{ij} = [\![i = j]\!]$
det	determinant of a matrix
e_j	the j^{th} canonical basis vector, the i^{th} element of which is $\delta_{i,j}$
$\mathbb{E}[y]$	the expectation of y, where y might be a random variable
$\circ$	Hadamard/Schur/element-wise product between vectors: $u \circ v = (u_1 v_1, \ldots, u_n v_n)^T$
$\mathcal{I}(\theta\|y)$ or $\mathcal{I}(\theta)$	information at θ (negative Hessian or approximate Hessian of log likelihood)
$\kappa(A)$	the (spectral) condition number of the matrix A; i.e., $\rho(A)/\rho(A^{-1})$
$\mathcal{K}_m(A, b)$ or $\mathcal{K}(A, b)$	the m^{th} Krylov subspace; span $\{b, Ab, A^2b, \ldots, A^{m-1}b\}$. When m isn't provided, assume it is such that the process generating the basis has terminated
$\ell(\theta\|y)$ or $\ell(\theta)$	log-likelihood evaluated θ having observed y

ln	(scalar) natural logarithm
log	principal matrix logarithm
$\succ$, $\succeq$	a matrix $A \succ 0$ is positive definite, $A \succ 0$ is positive-semidefinite
$\mathcal{MVN}(\mu, \Sigma)$	multivariate normal distribution with mean μ, variance/covariance Σ
$\mathcal{N}(\mu, \sigma^2)$	univariate normal distribution with mean μ, variance σ
$\|\cdot\|$	the Euclidean norm
$\|\cdot\|_A$	the norm associated with the inner product induced by $A \in \mathbb{S}^n_{++}$
$\mathcal{P}^m$	the space of polynomials with degree at most m
supp y	the support of the random variable y
$\mathbb{S}^m_{++}$	the space of matrices $A \in \mathbb{R}^{m \times m}$ such that $A = A^T \succ 0$. Analogously, $A \in \mathbb{S}^m_{+}$ implies that $A = A^T \succeq 0$
$\sigma(A)$	the spectrum of the matrix A
$\rho(A)$	the spectral radius of the matrix A
Tr	matrix trace
$\mathcal{V}(\theta\|y)$ or $\mathcal{V}(\theta)$	score function at θ (gradient of the log likelihood)

A.2 Abbreviations

AI	Average Information
BLAS	Basic Linear Algebra Subprograms
BLCG	Block Lanczos Conjugate Gradients
BLUP	Best Linear Unbiased Predictor
CG	Conjugate Gradients
GRM	Genomic Relatedness Matrix; the scaled correlation matrix $\frac{1}{m}ZZ^T$, where the columns of $Z \in \mathbb{R}^{n \times m}$ are the genotypes of n individuals at one of m markers, scaled to zero mean and unit variance
LCG	Lanczos Conjugate Gradients
MC	Monte Carlo
ML	Maximum Likelihood
NR	Newton Raphson

REML Restricted/Residual Maximum Likelihood

SBLLE Stochastic Block Lanczos Likelihood Estimation

Appendix B

Identities in multivariate statistics and matrix algebra

The following facts are used throughout the present thesis.

Fact B.1. $\det(\lambda A) = \lambda^N det(A)$ *for* $A \in \mathbb{F}^{N\times N}$, $\lambda \in \mathbb{F}$.

Fact B.2. *Let* V *be a non-singular matrix-valued function of a scalar* x. *Then*

$$\frac{\partial}{\partial\gamma}\ln(\det V) = Tr\,\left(V^{-1}\frac{\partial V}{\partial\gamma}\right).$$

Fact B.3. *Let* V *be a non-singular matrix-valued function of a scalar* γ. *Then*

$$\frac{\partial}{\partial\gamma}V^{-1} = -V^{-1}\frac{\partial V}{\partial\gamma}V^{-1}.$$

Fact B.4. *Let* $y \sim \mathcal{N}(\mu,\Sigma)$, *supp* $y = \mathbb{R}^N$. *Then the log likelihood of the parameters given the data is*

$$\ell(\mu,\Sigma|y) = -\frac{1}{2}\left(\ln(\det\Sigma) + (x-\mu)^T\Sigma^{-1}(x-\mu) + N\ln(2\pi)\right).$$

Fact B.5. *Let* Y, Z *be multivariate random vectors with joint distribution*

$$\begin{bmatrix} y \\ z \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix}\mu_y \\ \mu_z\end{bmatrix}, \begin{bmatrix}\Sigma_{yy} & \Sigma_{yz} \\ \Sigma_{zy} & \Sigma_{zz}\end{bmatrix}\right).$$

Note that Y *and* Z *need not take values in spaces of equal dimension. The conditional distribution of* $Z|Y$ *is given by*

$$(Z|Y=y) \sim \mathcal{N}\left(\mu_{z|y}, \Sigma_{z|y}\right)$$

where

$$\mu_{z|y} = \mu_z + \Sigma_{yz}\Sigma_{yy}^{\dagger}(y-\mu_y)$$

$$\Sigma_{z|y} = \Sigma_{zz} - \Sigma_{yz}\Sigma_{yy}^{\dagger}\Sigma_{zy}.$$

Fact B.6. *Let A be a symmetric matrix and let y be a random variable such that*

$$\mathbb{E}[y] = \mu \qquad Var(y) = \Sigma.$$

Then the expectation of the quadratic form $y^T Ay$ is given by

$$\mathbb{E}[y^T Ay] = Tr\ (A\Sigma) + \mu^T A\mu.$$

Fact B.7. *The derivative with respect to x of a quadratic form $(y - Ax)^T S (y - Ax)$, where S is symmetric, is given by*

$$\frac{\partial}{\partial x} (y - Ax)^T S (y - Ax) = -2A^T S(y - Ax).$$

Fact B.8. *The trace operator and derivative operator are exchangeable.*

Fact B.9 (Blockwise inversion via Schur complement)**.** *Let C be a nonsingular symmetric matrix defined block-wise by*

$$C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix},$$

and denote the Schur complements

$$C/C_{22} = C_{11} - C_{12}C_{22}^{-1}C_{21},$$
$$C/C_{11} = C_{22} - C_{21}C_{11}^{-1}C_{12}.$$

Provided C_{11} is nonsingular, C^{-1} can be expressed as

$$C^{-1} = \begin{bmatrix} C_{11}^{-1} + C_{11}^{-1}C_{21} (C/C_{11})^{-1} C_{12}C_{11}^{-1} & -C_{11}^{-1}C_{21} (C/C_{11})^{-1} \\ -(C/C_{11})^{-1} C_{12}C_{11}^{-1} & (C/C_{11})^{-1} \end{bmatrix},$$

or, equivalently, if C_{22} is non singular, C^{-1} can be expressed as

$$C^{-1} = \begin{bmatrix} (C/C_{22})^{-1} & -(C/C_{22})^{-1} C_{21}C_{22}^{-1} \\ -C_{22}^{-1}C_{12} (C/C_{22})^{-1} & C_{22}^{-1} + C_{22}^{-1}C_{12} (C/C_{22})^{-1} C_{21}C_{22}^{-1} \end{bmatrix}.$$

$$C^{-1} = \begin{bmatrix} Q^{-1} & -Q^{-1}C_{21}C_{22}^{-1} \\ -C_{22}^{-1}C_{12}Q^{-1} & C_{22}^{-1} + C_{22}^{-1}C_{12}Q^{-1}C_{21}C_{22}^{-1} \end{bmatrix}.$$

Appendix C

Software implementations

C.1 Libraries

The code provided assumes the following libraries:

imported libraries

```
from numpy import *
from numpy import linalg as npla
from scipy import linalg as la
from scipy import sparse
from scipy import special as sp
```

C.2 Spectral Newton Raphson

imported libraries

```
####### Spectral Newton Raphson ########
def ml_nr_spectral(
    start,      ## starting value for heritable component
    K,          ## relatedness matrix
    y,          ## phenotype vector
    ε = 1e-5,   ## tolerance
    maxIter = 10, verbose = False):

    Hess = zeros([2,2])
    Score = zeros(2)
    n = shape(K)[0]
    I = eye(n)
    j = 1
    printRate = maxIter // 5
    s2g = start

    ## perform spectral decomposition of K:
    D,Q = la.eigh(K)
    ## precompute matvecs:
    Qy = Q.T@y

    ## Newton iteration
    while j < maxIter:
        s2g0 = s2g
        s2e = 1 - s2g

        ## change of variables:
        τ = 1/s2e
        λ = s2g/s2e

        ## gradient in O(n)
        Score[0] = -.5*(-n/τ + (Qy**2).T @ (1/(λ*D+1)))
        Score[1] = -.5*(sum(D/(λ*D+1))-τ*(Qy**2).T @ (D/(λ*D+1)**2))

        ## Hessian in O(n)
```

```
        Hess[0,0] = n*τ**-2
        Hess[1,0] = Hess[0,1] = -(Qy**2).T @ (D/(λ*D+1)**2)
        Hess[1,1] = -sum(D**2 /(λ*D+1)**2) + 2*τ*(Qy**2).T @ (D**2 / (λ*D+1)**3)

        ## Newton update
        τ,λ = array([τ,λ]) + .5*la.solve(Hess,Score)

        ## change of variables:
        s2e = 1/τ
        s2g = 1-s2e

        ## check convergence:
        delta_s2g = abs(s2g - s2g0)
        if verbose and (j % printRate)==1:## starting value for heritable component
            print("step {}: s2g = {}, change = {}".format(j,round(s2g,5),round(delta_s2g,5)))
        if delta_s2g < ε:
            break
        else:
            j += 1

    return s2g
```

C.3 Monte Carlo Newton Raphson

MC Newton Raphson

```
####### Monte Carlo Newton Raphson ########
# follows https://doi.org/10.1038/ng.3190 #
def MCNR(
    GRM,            ## relatedness matrix
    Z,              ## genotype matrix
    y,              ## phenotype vector
    ε = 1e-5,       ## tolerance
    nMCtrials = 15, ## number of probes/trace apprx
    maxit = 100, verbose = True, maxSecant = 15):

    f = zeros(maxSecant)
    logTau = zeros(maxSecant)
    n = Z.shape[0]
    m = Z.shape[1]

    ## random latent u:
    u_rand = random.randn(m, nMCtrials)
    ## random latent e:
    e_rand_unsc = random.randn(n,nMCtrials)

    ## precompute matvec
    Zu_rand = Z @ u_rand

    ## starting value for secant iteration:
    h2 = sqrt(.25)
    τ = (1-h2)/h2
    logTau[0] = log(τ)

    ## construct objective function:
    def evalfMC(logTau):
        delta = exp(logTau)
        H = GRM + delta*eye(n)
        y_rand= Zu_rand + sqrt(delta) * e_rand_unsc
```

```
        ## solve linear systems to compute BLUPs
        HinvY_randT = array([spla.cg(H, y_rand[:,j])[0] for j in arange(nMCtrials)])
        blup_u_randT = array([Z.T @ HinvY_randT[j,:] for j in arange(nMCtrials)])
        blup_e_randT = array([delta* HinvY_randT[j,:] for j in arange(nMCtrials)])

        HinvY_data = spla.cg(H, y)[0]
        blup_u_data = Z.T @ HinvY_data
        blup_e_data = delta *  HinvY_data

        ## criterion for root finding:
        fMC = log(sum(blup_u_data**2)*sum(blup_e_randT**2)/(
            sum(blup_e_data**2)*sum(blup_u_randT**2)))

        return fMC

    ## initial iteration
    f[0] = evalfMC(logTau=logTau[0])
    if f[0] < 0:
        h2 = sqrt(.125)
    else:
        h2 = sqrt(.5)
    τ = (1-h2)/h2
    logTau[1] = log(τ)
    f[1] = evalfMC(logTau=logTau[1])

    ## additional secant iterations:
    J=2
    for j in arange(2,maxSecant):
        print("working on ",j)
        logTau[j] = (logTau[j-2]*f[j-1] - logTau[j-1]*f[j-2])/(f[j-1]-f[j-2])
        if abs(1/(1+exp(logTau[j]))-1/(1+exp(logTau[j-1]))) < ε:
            break
        else:
            f[j] = evalfMC(logTau=logTau[j])
            J+=1

    ## additional secant iterations:
    H = GRM + (exp(logTau[min(J,maxSecant-1)]))*eye(n)
    return y.T @spla.cg(H, y)[0] / n
```

C.4 Block Lanczos Conjugate Gradient update for seed systems (BLCG-seed)

BLCG for seed system

```
## constructs bases for krylov subspaces B, AB, AAB, ...##
def lanczos_cg_block(A, B, ε = 1e-9, p_freq = 5,
                     maxit = 100, verbose = True,
                     norm = lambda x: la.norm(x, 2)):
    n = A.shape[0]
    t = B.shape[1]

    X = zeros((n,t,maxit))
    R = zeros((n,t,maxit))
    U = zeros((n,t,maxit))

    ## coefficients
    ρ = zeros((t,maxit))
    β = zeros((t,maxit))
```

```
    ω = zeros((t,maxit))
    γ =  ones((t,maxit))
    δ = zeros((t,maxit))

    ## initial values
    ρ[:,0] = apply_along_axis(norm,0,B)
    β[:,0] = ρ[:,0]
    R[:,:,0] = U[:,:,0] = B
    U[:,:,0] = R[:,:,0]/β[:,0]

    j = 0
    cnvg = False

    while j < maxit-1:

        ## Lanczos iteration
        δ[:,j] = diag(U[:,:,j].T @ A @ U[:,:,j])
        tmp = A @ U[:,:,j] - U[:,:,j]*δ[:,j] - β[:,j]*U[:,:,j-1]
        β[:,j+1] = apply_along_axis(norm,0,tmp)
        U[:,:,j+1] = tmp / β[:,j+1]

        ## CG coefficents update
        γ[:,j] = (δ[:,j] - ω[:,j-1]/γ[:,j-1])**-1
        ω[:,j] = (β[:,j+1]*γ[:,j])**2
        ρ[:,j+1] = -β[:,j+1]*γ[:,j]*ρ[:,j]

        ## CG vectors update
        R[:,:,j+1] = ρ[:,j+1]*U[:,:,j+1]

        j += 1

        res_norm = norm(R[:,:,j])

        if verbose and j % p_freq ==0 : print("Error at step ",j,
                                    " is ", res_norm)
        if res_norm <= ε:
            cnvg = True
            break

    if verbose and cnvg: print("Converged after ",j," iterations.")
    elif verbose: print("Failed to converge after ",j," iterations.")

    return {
        "B":B,
        "U":U[:,:,0:j],
        "γ":γ[:,0:j],
        "β":β[:,0:j],
        "δ":δ[:,0:j],
        "ρ":ρ[:,0:j]
        }
```

C.5 Block Lanczos Conjugate Gradient update for shifted systems (BLCG-update)

BLCG update for shift

```
## solves (A + sigma I)X = B using results from lanczos_cg_block) ##
def lanczos_cg_update_block_real(seedSystem, σ, verbose= False,
                     ε = 1e-9, p_freq = 5,
```

```
                    norm = lambda x: la.norm(x, 2)):

    B = seedSystem["B"]
    U = seedSystem["U"]
    δ = seedSystem["δ"] + σ
    β = seedSystem["β"]
    ρ = seedSystem["ρ"]
    n = U.shape[0]
    t = U.shape[1]
    maxit = U.shape[2]

    X = zeros((n,t,maxit)) ## approximate soln
    P = zeros((n,t,maxit)) ## search directions
    R = zeros((n,t,maxit)) ## residual

    ## coefficients
    ω = zeros((t,maxit))
    γ =  ones((t,maxit))

    ## initial values
    R[:,:,0] = P[:,:,0] = B

    j = 0
    cnvg = False

    while j < maxit-1:

        γ[:,j] = (δ[:,j] - ω[:,j-1]/γ[:,j-1])**-1
        ω[:,j] = (β[:,j+1]*γ[:,j])**2
        ρ[:,j+1] = -β[:,j+1]*γ[:,j]*ρ[:,j]ʟ

        ## CG vectors update
        X[:,:,j+1] = X[:,:,j] + γ[:,j]*P[:,:,j]
        R[:,:,j+1] = ρ[:,j+1]*U[:,:,j+1]
        P[:,:,j+1] = R[:,:,j+1] + ω[:,j]*P[:,:,j]

        j += 1

        res_norm = norm(R[:,:,j])

        if verbose and j % p_freq ==0 : print("Error at step ",j,
                                  " is ", res_norm)
        if res_norm<=ε:
            cnvg = True
            break

    if verbose and cnvg: print("Converged after ",j," iterations.")
    elif verbose: print("Failed to converge after ",j," iterations.")

    return {
        "soln":X[:,:,j],
        "U":U[:,:,0:j],
        "γ":γ[:,0:j],
        "ω":ω[:,0:j],
        "ρ":ρ[:,0:j]
        }
```

C.6 Stochastic Lanczos quadrature (SLQ) to approximate log det from BLCG-seed output

SLQ to approximate log det from BLCG

```
##### Stochastic Lanczos Quadrature ######
### https://doi.org/10.1137/16M1104974 ###
def log_det_from_seed_SLQ(seeds, σ = 0, trueSoln = False, f = log):
    n_V = seeds['δ'].shape[0]
    Γ = 0

    for l in arange(0, n_V):
        θ, Y = la.eigh_tridiagonal(seeds["δ"][l,:] + σ,seeds["β"][l,1:])
        τ = Y[0,:]
        Γ += (τ**2) @ f(θ)

    trEst = n/n_V * Γ

    ## will give relative error if true soln is provided ##
    if trueSoln:
        return abs(trEst - trueSoln) /abs(trueSoln)
    else: return trEst
```

C.7 Stochastic Block Lanczos Likelihood Estimation (SBLLE)

SBLLE and auxilliary functions

```
def estimate_varcomp_BLCG_SLQ(
    A,                      ## GRM
    y,                      ## phenotype
    s2min=.01, s2max=.3,    ## bounds of heriatible component parameter space
    nShift =10,             ## number of nodes per "bi"-section iteration
    K=10,                   ## number of probe vectors for SLQ
    ε = 1e-4,               ## desired precision of VC estimate
    verbose=True):

    n = A.shape[0]                      ## problem dimsion
    h2 = linspace(s2max,s2min, nShift)  ## possible values of sigma**2
    τ  = array([(1-s2)/s2 for s2 in h2])## reparameterized
    τ0 = τ[0]                           ## seed value
    Σ  = [t - τ0 for t in τ]            ## shifts

    ## generate K normed Rademacher random vectors
    probes = array([(random.binomial(1,.5,size=n)*2 - 1) for k in arange(0,K)]).T
    probes = apply_along_axis(lambda x: x/la.norm(x),0,probes)

    ## construct Krylov bases for seed systems
    seeds = lanczos_cg_block_seed(A+τ0*I, hstack([y,probes]),verbose = verbose)

    ## initialize storage for eigendecompositions of Jacobi matrices
    Λ = zeros([K,seeds['δ'].shape[1]])  ## eigenvalues
    P = zeros([K,seeds['δ'].shape[1]])  ## first elements of eigenvectors

    ## initialize storage for solutions of linear systems and log det
    yAinvy = zeros(nShift)
    logDet = zeros(nShift)3741529

    ## compute eigendecompositions of Jacobi matrices
    for l in arange(0,K):
        Λ[l,:], tmpP  = la.eigh_tridiagonal(seeds["δ"][l+1,:], seeds["β"][l+1,1:])
```

```
        P[l,:]        = tmpP[0,:]

    j = 1 # counter
    if verbose:
        print("starting {}-section iteration {} over interval [{} -
        ↪  {}]".format(nShift,j,round(s2max,3), round(s2min,3)))

    ## compute expensive parts of the log likelihood
    for j in arange(0,nShift):
    σ = Σ[j]
    ## linear system:
    yAinvy[j] = y.T @  lanczos_cg_update_single_real(seeds, σ)
    ## log det:
    logDet[j] = log_det_from_eigen_SLQ(Λ, P, K, σ)

    ## compute log likelihoods
    log_like = -1 * (-n*log(τ+1) + logDet + (τ+1)*yAinvy)

    ## begin nShift-section optimization ##
    s2min, s2max = h2[flip(log_like.argsort())[1:3]] ## new search interval
    while abs(s2min-s2max) > ε:
        h2 = linspace(s2max,s2min, nShift)           ## new search nodes
        τ  = array([(1-s2)/s2 for s2 in h2])
        Σ  = [t - τ0 for t in τ]                     ## reparameterized as shifts
        j += 1
        if verbose:
            print("starting {}-section iteration {} over interval [{} -
            ↪  {}]".format(nShift,j,round(s2max,3),round(s2min,3)))
        ## compute expensive parts of the log likelihood
        for j in arange(0,nShift):
        σ = Σ[j]
        ## linear system:
        yAinvy[j] = y.T @  lanczos_cg_update_single_real(seeds, σ)
        ## log det:
        logDet[j] = log_det_from_eigen_SLQ(Λ, P, K, σ)
        ## compute log likelihoods:
        log_like = -1 * (-n*log(τ+1) + logDet + (τ+1)*yAinvy)
        ## choose new bounds for search interval:
        s2min, s2max = h2[flip(log_like.argsort())[1:3]]

    ## compute final estimate
    h2_estimate = h2[flip(log_like.argsort())[0]]
    log_like    = flip(sort(log_like))[0]
    if verbose:
    print("Converged to h2 ≈ {}".format(h2_estimate))

    return h2_estimate, log_like


### update SLQ trace estimate for shift from ###
### eigendecomposition of Jacobi matrix      ###
def log_det_from_eigen_SLQ(
        Λ, P,                  ## eigenvalues/vectors,
        n_V,                   ## number of probing vectors
        σ = 0,                 ## shift
        trueSoln = False,      ## for checking accuracy
        f = log):              ## function of matrix

    Γ = 0                      ## storage for approximate trace
```

```
    for l in arange(0, n_V):   ## perform SLQ
        τ = P[l,:]
        theta = Λ[l,:] + σ
        Γ += (τ**2) @ f(theta)

    trEst = n/n_V * Γ          ## estimate of Tr f(A)

    ## will give relative error if true soln is provided ##
    if trueSoln:
        return abs(trEst - trueSoln) /abs(trueSoln)
    else: return trEst

### solves only (A + sigma I)x = y using results ###
### using results from lanczos_cg_block_seed     ###
def lanczos_cg_update_single_real(
        seedSystem,     ## lanczos_cg_block_seed output
        σ,              ## shift
        verbose= False, ## detailed output
        ε = 1e-9,       ## tolerance
        p_freq = 5,     ## printing frequency
        norm = lambda x: la.norm(x, 2)):

    B = seedSystem["B"]
    n = B.shape[0]
    B = B[:,0]
    U = seedSystem["U"][:,0,:]
    δ = seedSystem["δ"][0,:] + σ
    ρ = seedSystem["ρ"][0,:]
    β = seedSystem["β"][0,:]
    n = U.shape[0]
    maxit = β.shape[0]

    X = zeros((n,maxit)) ## approximate soln
    P = zeros((n,maxit)) ## search directions
    R = zeros((n,maxit)) ## residual

    ## coefficients
    ω = zeros((maxit))
    γ =  ones((maxit))

    ## initial values
    R[:,0] = P[:,0] = B

    j = 0
    cnvg = False

    while j < maxit-1:

        γ[j] = (δ[j] - ω[j-1]/γ[j-1])**-1
        ω[j] = (β[j+1]*γ[j])**2
        ρ[j+1] = -β[j+1]*γ[j]*ρ[j]

        ## CG vectors update
        X[:,j+1] = X[:,j] + γ[j]*P[:,j]
        R[:,j+1] = ρ[j+1]*U[:,j+1]
        P[:,j+1] = R[:,j+1] + ω[j]*P[:,j]

        j += 1
```

```
        res_norm = norm(R[:,j])

        if verbose and j % p_freq ==0 : print("Error at step ",j,
                                    " is ", res_norm)
        if res_norm<=ε:
            cnvg = True
            break

    if verbose and cnvg: print("Converged after ",j," iterations.")
    elif verbose: print("Failed to converge after ",j," iterations.")

    return X[:,j].reshape(n,1)
```

C.8 Data simulation

Data Simulation

```
## simulate data
def simdata(
    n,     ## number of individuals
    m,     ## number of loci,
    h2=.1, ## true value of hertiable variance component
    verbose = False):

    ## variance component values:
    s0 = h2
    s1 = 1-s0

    ## allele frequencies and standardized genotypes:
    q = random.uniform(.1,.5,m)
    Z = array([random.binomial(1, p, n) for p in q]).T
    Z = apply_along_axis(lambda x: (x-mean(x))/(sqrt(m)*std(x)),1,Z)

    ## generate phenotype:
    I = eye(n)
    y = zeros([n,1])
    u = random.normal(0,1,m)
    y[:,0] = sqrt(s0) * Z@u + sqrt(s1)* random.normal(0,1,n)
    y[:,0] = (y[:,0] - mean(y[:,0]))/std(y[:,0])

    if verbose: print("var(Z@u) = {}, var(y) = ".format(std(Z@u)**2), std(y)**2)

    return Z @ Z.T,I,y,Z
```