Deflation techniques applied on mixed model equations

Vandenplas, Jeremie; Nguyen, Buu-Van; Vuik, Cornelis

DOI
10.1016/j.cam.2023.115095

Publication date
2023

Document Version
Final published version

Published in
Journal of Computational and Applied Mathematics

Citation (APA)
Vandenplas, J., Nguyen, B.-V., & Vuik, C. (2023). Deflation techniques applied on mixed model equations. *Journal of Computational and Applied Mathematics*, *426*, Article 115095. https://doi.org/10.1016/j.cam.2023.115095

Journal of Computational and Applied Mathematics 426 (2023) 115095

Contents lists available at ScienceDirect

Journal of Computational and Applied Mathematics

journal homepage: www.elsevier.com/locate/cam

Deflation techniques applied on mixed model equations

Jeremie Vandenplas [a,*], Buu-Van Nguyen [b], Cornelis Vuik [b]

[a] Animal Breeding and Genomics, Wageningen University & Research, P.O. Box 338, Wageningen, 6700 AH, The Netherlands
[b] Delft Institute of Applied Mathematics, Delft University of Technology, Mekelweg, 4, Delft, 2628 CD, The Netherlands

ARTICLE INFO

Article history:
Received 14 February 2022
Received in revised form 2 December 2022

Dataset link: https://doi.org/10.4121/19153742, https://git.wur.nl/vande018/vandenplas_dpcg_2022.git

Keywords:
Preconditioned conjugate gradient
Deflation
Proper orthogonal decomposition
K-means clustering
Genomic

ABSTRACT

In this paper, we consider a block Jacobi preconditioner and various deflation techniques applied in the Deflated Preconditioned Conjugate Gradient (DPCG) method for solving a sparse system of linear equations derived from a statistical linear mixed model that analyses simultaneously phenotypic and pedigree information of genotyped and ungenotyped animals with Single Polymorphism Nucleotide genotypes of genotyped animals. In livestock production systems, evaluating the genetic merit of the animals through such a model is a key process to ensure an improvement of animals for some characteristics of interest at each generation. First, we propose to define the deflation vectors using a subdomain deflation approach that considers some biological properties of the genotypes. Using simulated data, this approach reduces the number of iterations by up to 87% in comparison to a Preconditioned Conjugate Gradient method with a Jacobi preconditioner. Furthermore, compared to a DPCG method with same number of subdomains but defined randomly, this approach reduces the number of iterations by up to 20% for the same computational costs of one DPCG iteration. The properties of the resulting systems show that this approach annihilates the largest eigenvalues of the preconditioned coefficient matrix. Second, we propose the use of solution vectors of 12 systems of equations that include between 0.25% and 3% less data, as deflation vectors. For reducing the computational costs, we also consider a Proper Orthogonal Decomposition-reduced set of these 12 vectors. The properties of the resulting systems show that this recycling information approach annihilates the smallest eigenvalues of the preconditioned coefficient matrix, and results in a reduction of up to 39% in comparison to the PCG method. Finally, based on our experiment, the combination of the subdomain deflation approach relying on biological properties and of the POD-based approach to recycle previous solution vectors, for defining the deflation vectors, results in annihilating both the smallest and largest eigenvalues, and in a reduction of up to 88 % of the number of iterations in comparison to the PCG method.

1. Introduction

In livestock production systems, animal breeding has an important role because it ensures an improvement of animals at each generation through a selection process in order to meet breeding objectives that reflect the market demands. This selection process requires the ranking of the animals for the characteristics of interest, called traits (e.g., milk production). The best animals are thereafter selected to be parents of the next generation. Currently, evaluating and ranking animals is based on so-called breeding values, that are the average additive effects of the genes that an individual receives from both

* Corresponding author.
E-mail address: jeremie.vandenplas@wur.nl (J. Vandenplas).

https://doi.org/10.1016/j.cam.2023.115095

parents for a particular trait. Since the breeding values cannot be directly observed, they are estimated through a genetic evaluation based on three types of information: the performance records of the animals (hereafter called phenotypes), the pedigree, and since a decade, the Single Nucleotide Polymorphism (SNP) genotypes of a portion of the population [1,2].

The genetic evaluation of animals relies on a statistical linear model of the form [1]:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{W}\mathbf{u} + \mathbf{e} \tag{1}$$

where $\mathbf{y} \in \mathbb{R}^{n_p}$ is the vector of phenotypes, $\boldsymbol{\beta} \in \mathbb{R}^{n_\beta}$ is the vector of n_{fixed} fixed effects, $\mathbf{u} \in \mathbb{R}^{n_u}$ is the vector of random additive genetic effects (i.e. the breeding values), and $\mathbf{e} \in \mathbb{R}^{n_p}$ is the vector of random residual effects. The incidence matrices $\mathbf{X} \in \mathbb{R}^{n_p \times n_\beta}$ and $\mathbf{W} \in \mathbb{R}^{n_p \times n_u}$ relate the phenotypes to the fixed and random additive genetic effects, respectively. It is assumed that the expectations of the random effects are zero, that the residual effects are independently distributed, and that the variance–covariance matrix of the random additive genetic effects is a symmetric positive definite matrix proportional to a relationship matrix among all animals.

Before the advent of SNP genotypes at a large scale, the relationship matrix was traditionally computed only from the pedigree of the animals [1]. When genotypes of animals became available, animal breeders developed the so-called single-step genomic Best Linear Unbiased Prediction (ssGBLUP), in which the pedigree-based relationship matrix is replaced by a relationship matrix computed from both the pedigree and the SNP genotypes of the animals [2–4]. Because ssGBLUP can be easily implemented in existing software used in animal breeding, it quickly became the method of choice to analyse simultaneously phenotypic and pedigree information of genotyped and ungenotyped animals with genomic information of genotyped animals.

However, with the increasing amounts of genomic information, the initial ssGBLUP system of linear equations shows some computational limitations. For example, its solving needs the inversion of a dense symmetric genomic relationship matrix computed from the SNP genotypes of all genotyped animals and of size usually greater than 100,000 nowadays. Therefore, several approaches have been proposed to overcome these computational limitations, e.g., by approximating the inverse of the genomic relationship matrix [5], or by implicitly computing its inverse using the Woodbury matrix identity [6]. Some equivalent systems of equations that do not rely on a genomic relationship matrix, called hereafter single-step SNP BLUP (ssSNPBLUP), were also proposed in the literature [6–10]. These ssSNPBLUP approaches rely on the estimation of the effects of the SNP markers that compose the genotypes of the animals.

The ssSNPBLUP approaches are associated with systems of linear equations with sparse and symmetric positive (semi-)definite (SPSD) coefficient matrices. Because the Preconditioned Conjugate Gradient (PCG) method is commonly used in animal breeding for solving linear systems [11,12], the PCG method was the primarily choice for solving ssSNPBLUP systems. However, applying the PCG method to ssSNPBLUP systems showed convergence issues [10,13]. Vandenplas et al. [13] reported that the poor convergence behaviours of the PCG methods are associated with poor effective spectral condition numbers of the coefficient matrices of ssSNPBLUP preconditioned with a Jacobi preconditioner. In this study, the effective spectral condition number of a coefficient matrix $\mathbf{C} \in \mathbb{R}^{n_{eq} \times n_{eq}}$ is defined as the ratio between the largest and the smallest non-zero eigenvalue of $\mathbf{C}$ [14]. Using a Deflated PCG (DPCG) method with a subdomain deflation approach, these authors showed that the DPCG method applied to ssSNPBLUP resulted in a deflation of the largest eigenvalues of the preconditioned coefficient matrix, in a smaller effective spectral condition number, and therefore in faster convergence. However, their proposed definition for the deflation-subspace matrix may result in large and dense symmetric matrices for large genomic evaluations [13,15].

Based on these previous results, the first aim of this study is therefore to investigate different definitions of the subdomains that results in a more efficient DPCG method. Our second aim is to investigate the efficiency of recycling information from previous PCG methods applied on a subset of the data. Indeed, in practice, genomic evaluations are performed on a routinely basis, after the addition of phenotypic, genomic, and pedigree information, collected after the last genomic evaluation.

This work is divided into six sections. In Section 2, we describe the system of linear equations of ssSNPBLUP proposed by Gengler et al. [16] and by Liu et al. [7]. Section 3 proposes a brief description of the preconditioners and the DPCG method used in this study. In Section 4, we present various definitions of the deflation-subspace matrix proposed by Nguyen [17]. In Section 5, we describe the simulated data used in the numerical experiments and we present the results related to the efficiency of the different iterative methods using the simulated data. The last section reports our conclusions and recommendations.

2. Single-step genomic mixed model equations

In this study, we investigate different Preconditioned Conjugate Gradient-based methods for solving the ssSNPBLUP mixed model equations proposed by Gengler et al. [16] and by Liu et al. [7]. The ssSNPBLUP system of linear equations associated with Eq. (1) can be summarized as follows:

$$\mathbf{C}\mathbf{x} = \mathbf{b}, \tag{2}$$

where $\mathbf{C} \in \mathbb{R}^{n_{eq} \times n_{eq}}$ is a symmetric positive (semi-)definite coefficient matrix, $\mathbf{x} \in \mathbb{R}^{n_{eq}}$ is the vector of solutions, and $\mathbf{b} \in \mathbb{R}^{n_{eq}}$ is the right-hand side of the linear system.

For simplicity, and without loss of generality, the different matrices and vectors for the ssSNPBLUP system are described below for a univariate animal model. Following Liu et al. [7], the coefficient matrix $\mathbf{C}$ is equal to:

$$\mathbf{C} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}_n'\mathbf{R}_n^{-1}\mathbf{W}_n & \mathbf{X}_g'\mathbf{R}_g^{-1}\mathbf{W}_g & \mathbf{0} \\ \mathbf{W}_n'\mathbf{R}_n^{-1}\mathbf{X}_n & \mathbf{W}_n'\mathbf{R}_n^{-1}\mathbf{W}_n + \mathbf{A}^{nn}\sigma_u^{-2} & \mathbf{A}^{ng}\sigma_u^{-2} & \mathbf{0} \\ \mathbf{W}_g'\mathbf{R}_g^{-1}\mathbf{X}_g & \mathbf{A}^{gn}\sigma_u^{-2} & \mathbf{W}_g'\mathbf{R}_g^{-1}\mathbf{W}_g + (\mathbf{A}^{gg} + \boldsymbol{\Sigma}_{11})\,\sigma_u^{-2} & \boldsymbol{\Sigma}_{12}\sigma_u^{-2} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma}_{21}\sigma_u^{-2} & \boldsymbol{\Sigma}_{22}\sigma_u^{-2} \end{bmatrix},$$

where the subscripts g and n refer to n_g genotyped and n_n non-genotyped animals, respectively, the matrix $\mathbf{R}^{-1} = \begin{bmatrix} \mathbf{R}_n^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_g^{-1} \end{bmatrix}$ is the inverse of the diagonal residual (co)variance structure matrix, $\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}^{nn} & \mathbf{A}^{ng} \\ \mathbf{A}^{gn} & \mathbf{A}^{gg} \end{bmatrix}$ is the inverse of the pedigree-based relationship matrix of size $n_u = n_g + n_n$, σ_u^{-2} is the inverse of the additive genetic variance, the matrix $\boldsymbol{\Sigma}$ of size $n_g + n_s$ (with n_s being the number of SNP markers) is equal to $\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} = \begin{bmatrix} \left(\frac{1}{w} - 1\right)\mathbf{A}_{gg}^{-1} & -\frac{1}{w}\mathbf{A}_{gg}^{-1}\mathbf{Z} \\ -\frac{1}{w}\mathbf{Z}'\mathbf{A}_{gg}^{-1} & \frac{1}{w}\mathbf{Z}'\mathbf{A}_{gg}^{-1}\mathbf{Z} + \frac{m}{1-w}\mathbf{I} \end{bmatrix}$, the matrix $\mathbf{A}_{gg}$ is the pedigree-based relationship between genotyped animals, w is the proportion of variance (due to additive genetic effects) considered as residual polygenic effects, and $m = 2\sum p_o\,(1 - p_o)$ with p_o being the allele frequency of the oth SNP marker. The $n_g \times n_s$ matrix $\mathbf{Z}$ contains the SNP genotypes (coded as 0 for one homozygous genotype, 1 for the heterozygous genotype, or 2 for the alternate homozygous genotype) centred by their observed means.

The vector $\mathbf{x}$ is equal to $\mathbf{x} = \begin{bmatrix} \boldsymbol{\beta} \\ \mathbf{u}_n \\ \mathbf{u}_g \\ \mathbf{g} \end{bmatrix}$ where $\boldsymbol{\beta}$ is the vector of fixed effects, $\mathbf{u}_n$ is the vector of additive genetic effects for n_n non-genotyped animals, $\mathbf{u}_g$ is the vector of additive genetic effects for n_g genotyped animals, and $\mathbf{g}$ is the vector of n_s SNP effects.

Finally, the vector $\mathbf{b}$ is equal to $\mathbf{b} = \begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{y} \\ \mathbf{W}_n'\mathbf{R}_n^{-1}\mathbf{y}_n \\ \mathbf{W}_g'\mathbf{R}_g^{-1}\mathbf{y}_g \\ \mathbf{0} \end{bmatrix}$.

Assuming an univariate animal model with only one random effect (that corresponds to $\mathbf{u}$), an upper bound of the number of non-zero elements in the sparse matrix $\begin{bmatrix} \mathbf{X}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{X}'\mathbf{R}^{-1}\mathbf{W} \\ \mathbf{W}'\mathbf{R}^{-1}\mathbf{X} & \mathbf{W}'\mathbf{R}^{-1}\mathbf{W} \end{bmatrix}$ of size $n_\beta + n_u$ is equal to $n_p * \left(n_{fixed} + 1\right)^2$; an upper bound of the number of non-zero elements in the sparse matrix $\mathbf{A}^{-1}$ is equal to $7 * n_u$ [18]; and the number of non-zero elements in the dense matrix $\boldsymbol{\Sigma}$ is equal to $\left(n_g + n_s\right)^2$, because both the matrices $\mathbf{A}_{gg}^{-1}$ and $\mathbf{Z}$ are dense matrices [19]. Summing up the different upper bounds results in an upper bound of the number of non-zero elements in the coefficient matrix $\mathbf{C}$ equal to $n_p * \left(n_{fixed} + 1\right)^2 + 7n_u + \left(n_g + n_s\right)^2$. Values for n_p, n_{fixed}, n_u, n_g and n_s vary largely in practice. For example, by using the numbers provided in Vandenplas et al. [13] for a field dataset, the number of equations was larger than $6.45 * 10^6$, $n_p = 3,882,772$, $n_{fixed} = 7$, $n_u = 6,130,519$, $n_g = 90{,}963$, and $n_s = 37{,}995$, resulting in an upper bound of the number of non-zero elements of the coefficient matrix $\mathbf{C}$ for a single trait larger than $16.9 * 10^9$ and in a density of $\mathbf{C}$ of approximately 0.05%.

3. Iterative solvers

The system of linear equations (2) is usually solved with the PCG method with a Jacobi preconditioner [10–13]. However, the PCG method applied to the ssSNPBLUP system results in convergence issues. Recently, Vandenplas et al. [13,15] showed that a DPCG method can improve the convergence behaviour. Similar convergence improvements were also achieved with a block-diagonal Jacobi preconditioner [20] with a dense matrix of size n_s. In this section, we give an overview of the different preconditioners and of the DPCG method used in this study.

3.1. PCG method and preconditioners

The PCG method consists in transforming the linear system (2) with the inverse of a symmetric positive definite matrix $\mathbf{M}$, called preconditioner, into an equivalent system of linear equations for which the resulting system matrix, $\mathbf{M}^{-1}\mathbf{C}$, called the preconditioned coefficient matrix, has an effective spectral condition number, $\kappa(\mathbf{M}^{-1}\mathbf{C})$, smaller than the effective spectral condition number of $\mathbf{C}$, κ ($\mathbf{C}$) [14,21]. Briefly, the PCG method generates a sequence of solution vectors $\mathbf{x}^k$, such

that the error of $\mathbf{x}^k$ is bounded by [21]:

$$\|\mathbf{x}-\mathbf{x}^k\|_C \leq 2\left(\frac{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{C})}-1}{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{C})}+1}\right)^k \|\mathbf{x}-\mathbf{x}^0\|_C \tag{3}$$

where $\mathbf{x}^0$ is a starting solution vector, $\mathbf{r}^0$ is the residual vector defined as $\mathbf{r}^0 = \mathbf{b} - \mathbf{C}\mathbf{x}^0$.

In animal breeding, matrix-free iterative solvers are commonly used [11–13]. Therefore, diagonal preconditioners, also called Jacobi preconditioners and defined as $\mathbf{M} = diag(\mathbf{C})$, are preferred because they can be easily constructed and their associated systems of equations are easy to solve [11–13,22]. More complex models that involve multiple traits or correlated effects will require block Jacobi preconditioners with blocks of small dimensions (typically of size equal to the number of traits or correlated effects) [23].

The first implementations of the PCG solvers with a Jacobi preconditioner applied to ssSNPBLUP systems showed convergence issues [10,13]. Therefore, Konstantinov and Goddard [20] proposed the following block Jacobi preconditioner to improve the convergence behaviour of the PCG method applied to the ssSNPBLUP system:

$$\mathbf{M}_k = \begin{bmatrix} diag\,(\mathbf{C}_{oo}) & \mathbf{0} \\ \mathbf{0} & \left(\frac{1}{w}\mathbf{Z}'\mathbf{A}_{gg}^{-1}\mathbf{Z} + \frac{m}{1-w}\mathbf{I}\right)\sigma_u^2 \end{bmatrix}$$

where $\mathbf{C}_{oo}$ contains the diagonal elements of $\mathbf{C}$ corresponding to the equations of $\boldsymbol{\beta}$, $\mathbf{u}_n$, and $\mathbf{u}_g$.

While Konstantinov and Goddard [20] reported acceptable convergence behaviours, some limitations for applying $\mathbf{M}_k$ in large evaluations can be the time required for its computation, especially in matrix-free solvers using compressed genotype matrices $\mathbf{Z}$ (e.g., [18]), and its size with large numbers of SNP markers.

3.2. Deflated preconditioned conjugate gradient method

Deflation methods aim to annihilate the effect of unfavourable eigenvalues on the convergence of the PCG method, resulting in Deflated Preconditioned Conjugate Gradient (DPCG) methods [24,25].

Given a full rank deflation-subspace matrix, $\mathbf{Z}_d \in \mathbb{R}^{n_{eq}\times m_d}$ with $m_d < n_{eq} - d$ where d is equal to $\dim(\mathcal{N}(\mathbf{C}))$, the deflation matrix $\mathbf{P} \in \mathbb{R}^{n_{eq}\times n_{eq}}$ is defined as:

$$\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{Q} \tag{4}$$

where the matrix $\mathbf{Q} \in \mathbb{R}^{n_{eq}\times n_{eq}}$ is defined as $\mathbf{Q} = \mathbf{Z}_d\mathbf{E}^{-1}\mathbf{Z}_d^T$, with the matrix $\mathbf{E} \in \mathbb{R}^{m_d\times m_d}$ being the Galerkin matrix defined as $\mathbf{E} = \mathbf{Z}_d^T\mathbf{C}\mathbf{Z}_d$. The columns of the deflation-subspace matrix $\mathbf{Z}_d$ are called deflation vectors and are chosen such that $\mathbf{E}$ is an invertible matrix. Hence the matrix $\mathbf{Z}_d$ is chosen such that $\mathcal{N}(\mathbf{C}) \nsubseteq \mathcal{R}(\mathbf{Z}_d)$[26].

The deflation method can be applied to the preconditioned system $\mathbf{M}^{-1}\mathbf{C}\mathbf{x} = \mathbf{M}^{-1}\mathbf{b}$, as follows:

$$\mathbf{M}^{-1}\mathbf{P}\mathbf{C}\mathbf{x} = \mathbf{M}^{-1}\mathbf{P}\mathbf{b} \tag{5}$$

The algorithm of the DPCG method used for solving the linear system (5) is described in Algorithm 1 [27]. The error upper bound of the DPCG method is given by [27]:

$$\|\mathbf{x}-\mathbf{x}^k\|_C \leq 2\left(\frac{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{P}\mathbf{C})}-1}{\sqrt{\kappa(\mathbf{M}^{-1}\mathbf{P}\mathbf{C})}+1}\right)^k \|\mathbf{x}-\mathbf{x}^0\|_C \tag{6}$$

Therefore, as for to the PCG method, the convergence depends on the effective spectral condition number of $\mathbf{M}^{-1}\mathbf{P}\mathbf{C}$, $\kappa(\mathbf{M}^{-1}\mathbf{P}\mathbf{C})$.

Algorithm 1: Deflated Preconditioned Conjugate Gradient method for solving $\mathbf{C}\mathbf{x} = \mathbf{b}$

Choose $\mathbf{x}^0$
$\mathbf{r}^0 = \mathbf{b} - \mathbf{C}\mathbf{x}^0$
$\hat{\mathbf{r}} = \mathbf{P}\mathbf{r}^0$
Solve $\mathbf{M}\mathbf{z}^0 = \hat{\mathbf{r}}^0$
$\mathbf{p}^0 = \mathbf{z}^0$
for $k = 0, 1, \ldots,$ *until convergence* **do**
  $\hat{\mathbf{w}}^k = \mathbf{P}\mathbf{C}\mathbf{p}^k$
  $\alpha_k = \frac{<\hat{\mathbf{r}}^k, \mathbf{z}^k>}{<\mathbf{p}^k, \hat{\mathbf{w}}^k>}$
  $\hat{\mathbf{x}}^{k+1} = \hat{\mathbf{x}}^k + \alpha_k\mathbf{p}^k$
  $\hat{\mathbf{r}}^{k+1} = \hat{\mathbf{r}}^k - \alpha_k\hat{\mathbf{w}}^k$
  Solve $\mathbf{M}\mathbf{z}^{k+1} = \hat{\mathbf{r}}^{k+1}$
  $\beta_k = \frac{<\hat{\mathbf{r}}^{k+1}, \mathbf{z}^{k+1}>}{<\hat{\mathbf{r}}^k, \mathbf{z}^k>}$
  $\mathbf{p}^{k+1} = \mathbf{z}^{k+1} + \beta_k\mathbf{p}^k$
$\mathbf{x}_{final} = \mathbf{Q}\mathbf{b} + \mathbf{P}^T\hat{\mathbf{x}}^{k+1}$;

4. Computation of the deflation-subspace matrix

The deflation vectors of the deflation-subspace matrix $\mathbf{Z}_d$ can be defined following several techniques based on, e.g., approximated eigenvectors [28], recycling information of previous Krylov subspaces [25], or subdomain deflation vectors [29]. In this study, we investigate the techniques based on subdomain deflation and on recycling information of previous Krylov subspaces.

4.1. Subdomain deflation

For solving ssSNPBLUP systems, Vandenplas et al. [13] defined the deflation vectors following a subdomain deflation approach. Their proposed approach relies on grouping SNP effects in non-overlapping subdomains to annihilate the effect of the largest unfavourable eigenvalues of the preconditioned coefficient matrix on the convergence. Briefly, the ssSNPBLUP domain of a univariate system is divided as follows: (1) all fixed and random effects other than the SNP effects are included in a separate subdomain, and (2) each set of a fixed amount of randomly chosen (without replacement) SNP effects are included in the same subdomain. Therefore, each deflation vector, that is each column of $\mathbf{Z}_d$, which corresponds to a subdomain, contains values of 1 for the entries associated with an equation included in the corresponding subdomain, and 0 otherwise. Following this definition, each row of $\mathbf{Z}_d$ contains only one non-zero element, and each column of $\mathbf{Z}_d$ contains as many non-zero elements as the amount of equations associated with the corresponding subdomain. For multi-trait ssSNPBLUP systems, this division is applied within each trait (see Vandenplas et al. [13] for more details).

Based on this proposed approach, multiple divisions of the computational domain of the ssSNPBLUP system into a same amount of non-overlapping subdomains are possible. Following Vuik et al. [27], the optimal division may depend on the properties of the system of linear equations. For example, Vuik et al. [27] defined the subdomains based on the properties of the eigenvectors associated with the smallest eigenvalues of $\mathbf{M}^{-1}\mathbf{C}$ for a class of layered problems with extreme contrasts in $\mathbf{C}$. In this study, we investigate the potential benefit of a division of the computational domain of ssSNPBLUP based on biological properties of the SNP genotypes. Indeed, the associations between alleles of SNP markers in a population are not completely associated randomly due to many factors (e.g., population structure, selection). This non-random association of alleles at different SNP markers in a population is called linkage disequilibrium, and the level of linkage disequilibrium among all SNP markers can be approximated by the correlation matrix associated with the centred genotype matrix $\mathbf{Z}$ [30]. Assuming that correlated SNP markers are associated with similar estimated SNP effects (in sign and value), the division of the ssSNPBLUP domain based on groups of highly correlated SNP markers might be more appropriate than a division of the ssSNPBLUP domain based on groups of random SNP effects, as proposed by Vandenplas et al. [13]. In this study, the K-means++ algorithm [31] is applied on the correlation matrix associated with $\mathbf{Z}$ to define clusters of SNP markers associated with the same subdomain. The number of clusters is determined such that the total amount of subdomains is the same as the amount of subdomains based on the approach assigning SNPs to a subdomain randomly.

4.2. Recycling information from previous ssSNPBLUP evaluations

In genetic evaluation centers, genomic evaluations using models such as the ssSNPBLUP models are performed on a routinely basis (e.g., each 3–4 months in a dairy cattle context). Each genomic evaluation includes phenotypes, genotypes, and pedigree of the previous genomic evaluation, as well as information collected after this previous genomic evaluation. However, while the system of equations is growing at each new evaluation, the newly collected information represent only a small part of all the datasets considered by the system (usually less than 1%). Therefore, solution vectors obtained from previous genomic evaluations could be recycled as initial solution vector for a PCG-based algorithm, or as snapshots for defining a deflation-subspace matrix $\mathbf{Z}_d$ [32].

It is worth noting that the addition of new information to a genomic evaluation will generate additional equations that were not included in previous genomic evaluations (e.g., for animals born after the last genomic evaluation). In this study, this issue is avoided by assuming a same set of levels (e.g., animals) for all fixed and random effects. For the genetic effects, this is equivalent to the replacement of a missing entry corresponding to an animal's genomic breeding value in the solution vector by the mean of its parents' genomic breeding values.

4.3. Proper orthogonal decomposition deflation vectors

Following Diaz Cortes et al. [32], deflation vectors can be computed by applying the proper orthogonal decomposition (POD) method to a set of snapshots. In this study, the snapshots are solution vectors of previous ssSNPBLUP systems. The POD method applied to a set of snapshots generates a small set of orthonormal basis vectors $\{\phi_1, \phi_2, \ldots, \phi_p\}$ with $\phi_i \in \mathbb{R}^{n_{eq}}$ and $p \in \mathbb{N}$. The basis vectors ϕ_i are eigenvectors that correspond to the p largest eigenvalues of the data matrix defined as:

$$\mathbf{R}_s = \frac{1}{s-1}\mathbf{S}\mathbf{S}^T$$

where $\mathbf{S} = \begin{bmatrix}\hat{\mathbf{x}}_1 & \hat{\mathbf{x}}_2 & \ldots & \hat{\mathbf{x}}_s\end{bmatrix} \in \mathbb{R}^{n_{eq}\times s}$ with $\hat{\mathbf{x}}_i \in \mathbb{R}^{n_{eq}}$ being the ith snapshot.

In practice, the eigenvectors of $\mathbf{R}_s$ can be easily obtained by applying a singular value decomposition on the matrix $\frac{1}{s-1}\mathbf{S}^T\mathbf{S}$, which is a matrix of a much smaller size than $\mathbf{R}_s$, as detailed by Diaz Cortes et al. [32]. In this study, the two first vectors that contain the largest part of the variability of the snapshots are chosen as deflation vectors.

5. Numerical experiments

In this section, we present briefly the simulated dataset used for generating a ssSNPBLUP linear system. Then, details and performances of each approach used for solving the ssSNPBLUP system are provided.

5.1. Data

Datasets are simulated using the QMSim software [33], that is a software for simulating livestock pedigree, SNP genotypes and phenotypes. The parameter file needed for QMSim is the one provided by Bradford et al. [34]. Briefly, this simulation aims to mimic a dairy cattle population under selection for a female-limited trait associated with a heritability of 0.30. The final datasets include 164,500 animals in the pedigree, among which 18,678 animals are genotyped for 13,100 SNP markers on 5 chromosomes. A total of 12,446 SNP markers with a minor allele frequency higher than 0.1 are retained for ssSNPBLUP after filtering. A total of 82,440 phenotypes for only female individuals are available. More details on the simulation procedure can be found in Bradford et al. [34]. These pedigree, genotype, and phenotype datasets are referred below to as full datasets, and result in a ssSNPBLUP system with 176,947 equations and a density of $\mathbf{C}$ equal to 2.9%.

To simulate a real scenario in which datasets are incremented routinely, reduced phenotype and genotype datasets are created by removing between 0.25% and 3.00% of most recent records, by step of 0.25%. Therefore, a total of 12 phenotype and genotype reduced datasets are generated from the full datasets.

5.2. Implementation

The calculations are performed with the programming language *Julia* Version 1.7.3 [35]. The Julia package *Clustering.jl* (https://juliastats.org/Clustering.jl/stable/) is used to apply the K-means++ algorithm on the correlation matrix associated with $\mathbf{Z}$. The Julia package *TimerOutputs.jl*
(https://github.com/KristofferC/TimerOutputs.jl) is used to report wall clock times of different sections. The scripts to generate the datasets and the analyses are available at https://doi.org/10.4121/19153742 and at WUR Gitlab https://git.wur.nl/vande018/vandenplas_dpcg_2022.git.

5.3. Evaluations

For all the experiments, the ssSNPBLUP system (2) is set up using the full datasets, and a Jacobi preconditioner is used in all the cases.

For all iterative methods, the termination criterion is the relative residual defined as follows:

$$\frac{\|\mathbf{r}^k\|}{\|\mathbf{b}\|} \leq 10^{-6} \tag{7}$$

with $\|\mathbf{r}^k\|$ being the 2-norm of the residual of the kth iteration.

For all ssSNPBLUP systems, the smallest and largest Ritz values are computed with the Lanczos algorithm based on information obtained from the (D)PCG methods [21,36,37]. These extremal Ritz values are used for estimating the smallest and largest non-zero eigenvalues that influence the convergence of the (D)PCG methods, as well as the effective spectral condition number of the (deflated) preconditioned coefficient matrices.

All computations were performed on a cluster with nodes with 376 GB and an Intel Xeon Gold 6130 (2.10 GHz) processor with 32 cores.

5.3.1. The PCG method with a Jacobi preconditioner

The PCG method with a Jacobi preconditioner is commonly used in animal breeding to solve iteratively linear systems [11,12]. Therefore, this is considered as a reference method in this study for solving iteratively a ssSNPBLUP system.

The number of iterations to reach the termination criterion and Ritz values of the ssSNPBLUP preconditioned coefficient matrix, are in Table 1. The number of iterations to reach convergence is equal to 1742 with a zero initial solution vector, and reduces by 36% when the initial solution vector is the solution vector of a ssSNPBLUP evaluation for which the latest 0.25% of genomic and phenotypic information were removed (Fig. 1; Table 1). For both evaluations, the smallest and largest Ritz values are about $1.6 * 10^{-3}$ and 276.6, leading to an estimated effective spectral condition number of around $1.69 * 10^6$.

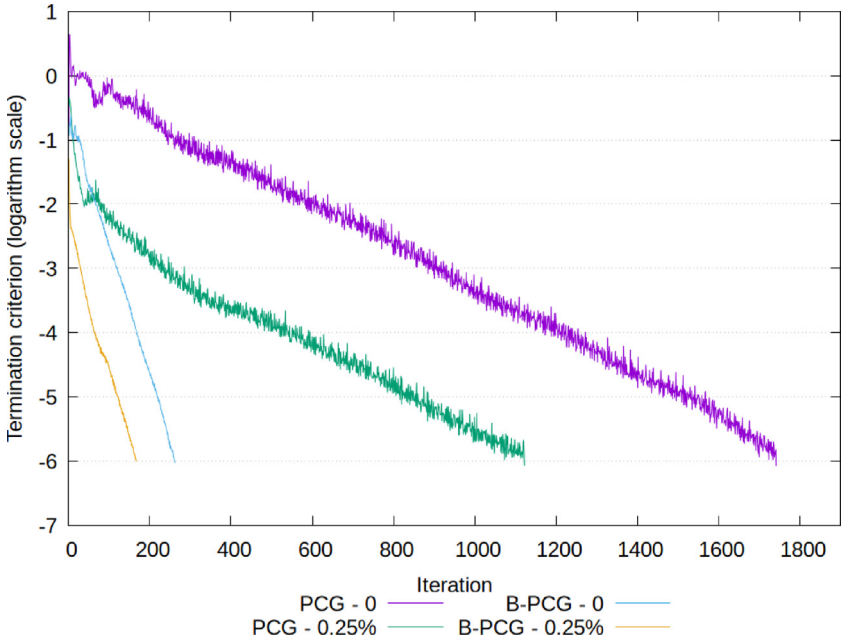

Fig. 1. Convergence plots for the PCG method with a Jacobi preconditioner (PCG) or a block Jacobi preconditioner (B-PCG), and with a zero initial solution vector (0) or with a solution vector from a previous evaluation (0.25%).

Table 1
Number of iterations to reach the termination criterion and Ritz values of different PCG-based approaches applied on a ssSNPBLUP linear system.

Approach[a]	Deflation	# def. vect.	Init. sol.	# iter.	Smallest Ritz	Largest Ritz
PCG	–	–	0	1742	$1.635 * 10^{-3}$	276.633
PCG	–	–	0.25%	1122	$1.685 * 10^{-3}$	276.633
PCG-B	–	–	0	262	$1.447 * 10^{-3}$	2.622
PCG-B	–	–	0.25%	167	$1.492 * 10^{-3}$	2.622
DPCG	Random	1246	0	376	$1.734 * 10^{-3}$	7.627
DPCG	Random	126	0	1113	$1.709 * 10^{-3}$	80.279
DPCG	Random	1246	0.25%	240	$1.717 * 10^{-3}$	7.627
DPCG	Random	126	0.25%	727	$1.701 * 10^{-3}$	80.279
DPCG	K-means	1246	0	337	$1.743 * 10^{-3}$	6.760
DPCG	K-means	126	0	890	$1.710 * 10^{-3}$	45.440
DPCG	K-means	1246	0.25%	217	$1.722 * 10^{-3}$	6.760
DPCG	K-means	126	0.25%	580	$1.702 * 10^{-3}$	45.440
DPCG	Snapshots	12	0	1057	$1.809 * 10^{-3}$	276.491
DPCG	Snapshots	12	0.25%	1052	$1.810 * 10^{-3}$	276.491
DPCG	POD	2	0	1215	$2.001 * 10^{-3}$	276.605
DPCG	POD	2	0.25%	1029	$1.761 * 10^{-3}$	276.605
DPCG	K-means + POD	1248	0.25%	207	$1.785 * 10^{-3}$	6.760
DPCG	K-means + POD	128	0.25%	563	$1.765 * 10^{-3}$	45.440

[a]PCG-B = PCG with a block Jacobi preconditioner; DPCG = deflated PCG.

5.3.2. The PCG method with a block Jacobi preconditioner

In this study, the block Jacobi preconditioner $\mathbf{M}_k$ [20] is also tested. In our implementation, a Bunch–Kaufman factorization [38] of the dense matrix $\left(\frac{1}{w}\mathbf{Z}'\mathbf{A}_{gg}^{-1}\mathbf{Z} + \frac{m}{1-w}\mathbf{I}\right)\sigma_u^2$ included in $\mathbf{M}_k$ is performed and stored before starting the PCG iterative process for allowing an efficient solving of the associated system of equations.

With $\mathbf{M}_k$, the number of iterations to reach convergence is equal to 262 with a zero initial solution vector, and to 167 with an initial solution vector being the solution vector of a previous evaluation with 0.25% less genotypes and phenotypes (Fig. 1; Table 1). For both scenarios, this is a reduction of 85% of the number of iterations of the corresponding PCG method with a Jacobi preconditioner. This reduction is also reflected mainly by a reduction of the largest Ritz values that become equal to 2.6 in both scenarios (Table 1).

5.3.3. The DPCG method based on a subdomain deflation approach

In this study, two approaches for defining the deflation vectors are investigated. The first approach consists of grouping randomly the SNP effects in non-overlapping subdomains of same size, as proposed by Vandenplas et al. [13,15]. The

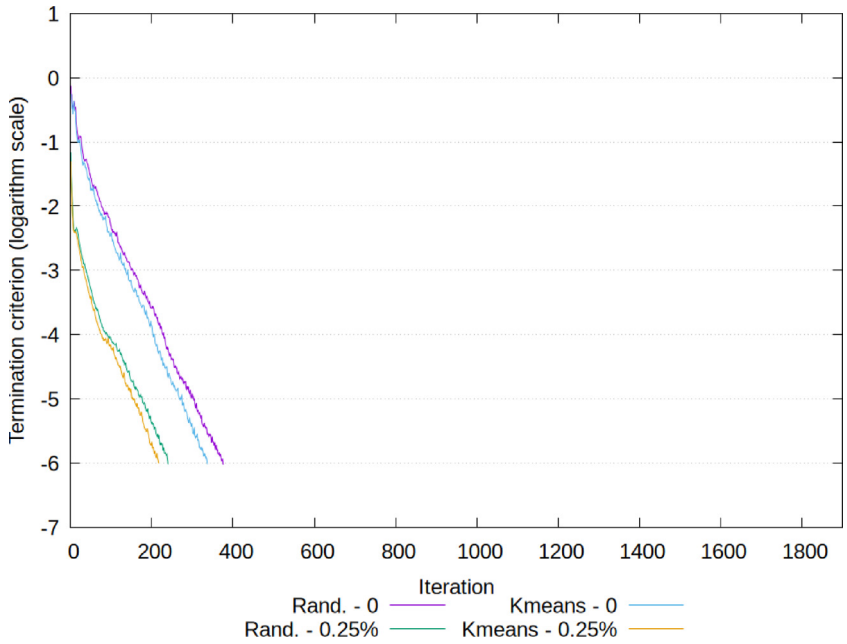

Fig. 2. Convergence plots for the DPCG method with 1246 deflation vectors, with a zero initial solution vector (0) or with a solution vector from a previous evaluation (0.25%). The deflation vectors were defined based on a subdomain deflation approach with subdomains defined randomly (Rand.) or based on a K-means++ algorithm (Kmeans).

second approach consists of grouping the SNP effects in non-overlapping subdomains based on the clusters defined by the K-means++ algorithm applied on the correlation matrix associated with $\mathbf{Z}$, as explained in Section 3. For evaluating the efficiency of the K-means algorithm-based approach over the random assignments approach, the number of clusters to be determined by the K-means++ algorithm is set to the number of subdomains obtained with the approach based on random assignments. In this study, a Jacobi preconditioner is always used in the DPCG method.

Using the random assignments approach, 10 and 100 SNP effects are randomly assigned to the same subdomain, resulting in 1245 and 125 subdomains associated with SNP effects, respectively. For the K-means algorithm-based approach, a total of 1245 and 125 clusters of SNPs are defined using the K-means++ algorithm, and each cluster corresponds to a subdomain associated with SNP effects. The first set of 1245 subdomains includes on average 10 SNP effects, with the smallest subdomain of this division being associated with only 1 SNP effect, and the largest subdomain being associated with 118 SNP effects. Similarly, the second set of 125 subdomains includes on average 100 SNP effects per subdomain, with at least 21 SNP effects per subdomain and at most 700 SNP effects per subdomain. Finally, by considering the subdomain associated to all effects other than the SNP effects, a total of 1246 and of 126 subdomains are defined using both the random assignments and the K-means algorithm-based approaches. It is worth noting that the computational cost for one DPCG iteration is the same with both subdomain definitions when using the same number of subdomains, because the deflation-subspace matrices have the same size and same number of non-zero elements for both approaches.

With 126 subdomains and a zero vector as initial solution vector, the number of iterations to reach convergence is equal to 1113 with the random assignments approach and reduces by about 20% when the K-means algorithm-based approach is used (i.e., 890 iterations; Figs. 2–3; Table 1). This confirms our expectation, also stated in Vandenplas et al. [13], that a definition of the subdomains based on properties of the genomic information can lead to a more efficient DPCG method with the same computational costs per iteration.

Increasing the number of subdomains to 1246 results in an additional decrease of the number of iterations to reach convergence. Indeed, only 376 iterations with the random assignments approach and 337 iterations with the K-means algorithm-based approach are needed to reach convergence (Figs. 2–3; Table 1). The smaller difference in terms of iterations between the two approaches can be explained by the fact that smaller subdomains yield to similar deflation vectors defined by both approaches.

Similarly to the PCG solver, using the solution vector of a previous evaluation with 0.25% less genotypes and phenotypes as initial solution vector, results in around 35% less iterations to reach convergence for both approaches and for all the scenarios (Figs. 2–3; Table 1).

Regarding the Ritz values, the smallest Ritz values of the deflated preconditioned coefficient matrices are about $1.7 * 10^{-3}$ for all DPCG approaches, and similar to the smallest Ritz values obtained from the PCG methods. However, variations of the largest Ritz values are observed across the different approaches. Indeed, the largest Ritz value decreases from 276.6 with the PCG method to 80.3 with the DPCG method and 126 subdomains defined with the random assignments approach.

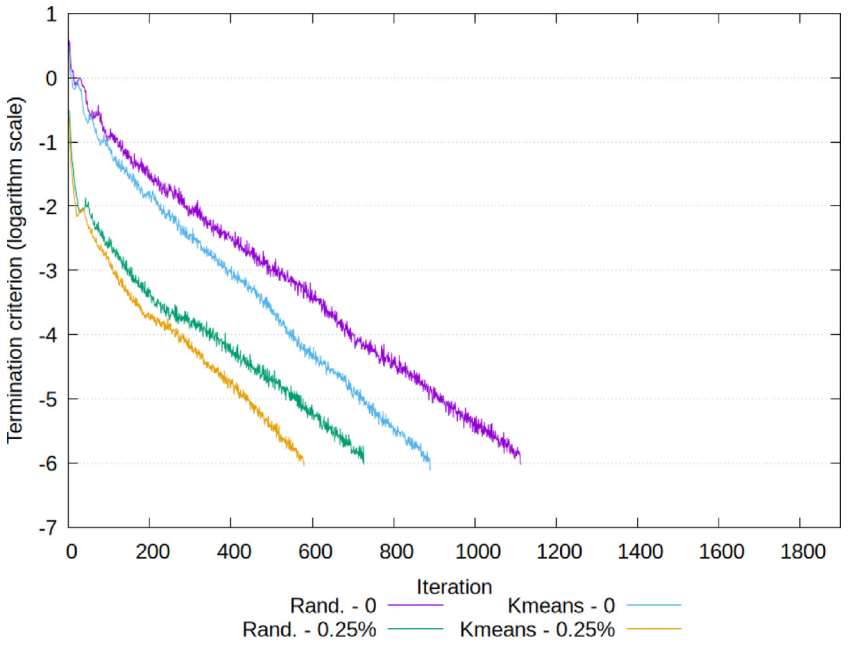

Fig. 3. Convergence plots for the DPCG method with 126 deflation vectors, with a zero initial solution vector (0) or with a solution vector from a previous evaluation (0.25%). The deflation vectors were defined based on a subdomain deflation approach with subdomains defined randomly (Rand.) or based on a K-means++ algorithm (Kmeans).

Refining the definition of the 126 subdomains with the K-means algorithm-based approach leads to a reduction of the largest Ritz value to 45.4. The increase of the number of subdomains results in additional reductions of the largest Ritz values, i.e. to 7.6 with the random assignments approach and to 6.8 with the K-means algorithm-based approach (Table 1).

5.3.4. The DPCG method based on recycling information from previous systems

As described in Section 3, genomic evaluations are performed routinely, with the addition of newly collected information at each evaluation. To simulate such a real scenario, solution vectors for 12 ssSNPBLUP systems are computed using reduced phenotype and genotype datasets obtained by removing from the full phenotype and genotype datasets between 0.25% and 3.00% of the most recent records.

First, the 12 solution vectors are used as snapshots for defining a deflation-subspace matrix $\mathbf{Z}_d$. The resulting DPCG method with a zero vector as initial solution vector converges in 1057 iterations. Using the solution vector of a ssSNPBLUP system with 0.25% data resulted in a similar number of iterations to reach convergence (i.e., 1052 iterations; Fig. 4; Table 1). Following Lemma 5.1, the number of iterations to reach convergence should be the same for both methods, and the difference observed in the number of iterations could be due to floating point errors.

Lemma 5.1. *Let* $\mathbf{C} \in \mathbb{R}^{n_{eq} \times n_{eq}}$ *be a symmetric positive semi-definite (SPSD) matrix,* $\mathbf{Z}_d \in \mathbb{R}^{n_{eq} \times m_d}$ *be a full rank deflation-subspace matrix and chosen such that* $\mathcal{N}(\mathbf{C}) \nsubseteq \mathcal{R}(\mathbf{Z}_d)$*, and the vector* $\mathbf{y} \in \mathbb{R}^{n_{eq}}$ *be a linear combination of the deflation vectors of* $\mathbf{Z}_d$*, defined as* $\mathbf{y} = \mathbf{Z}_d\mathbf{w}$ *with* $\mathbf{w} \in \mathbb{R}^{m_d}$*. The invertible Galerkin matrix* $\mathbf{E} \in \mathbb{R}^{m_d \times m_d}$ *and the deflation matrix* $\mathbf{P} \in \mathbb{R}^{n_{eq} \times n_{eq}}$ *are defined as* $\mathbf{E} = \mathbf{Z}_d'\mathbf{C}\mathbf{Z}_d$ *and* $\mathbf{P} = \mathbf{I} - \mathbf{C}\mathbf{Z}_d\mathbf{E}^{-1}\mathbf{Z}_d'$*, respectively [26]. Using* $\mathbf{x}^0 = \mathbf{y}$ *as initial solution vector in Algorithm 1 results in the same iterative process of a DPCG method using* $\mathbf{x}^0 = \mathbf{0}$ *as initial solution vector.*

Proof. Since $\mathbf{P}\mathbf{C}\mathbf{Z}_d = \mathbf{0}$, it follows that for any vector $\mathbf{y} = \mathbf{Z}_d\mathbf{w}$:

$$\mathbf{P}\mathbf{C}\mathbf{y} = \mathbf{P}\mathbf{C}\mathbf{Z}_d\mathbf{w} = \mathbf{0}\mathbf{w} = \mathbf{0}$$

Using this equality, it can be easily proved that the iterative process of the DPCG algorithm is the same when using the initial solution vector $\mathbf{x}^0 = \mathbf{y}$ or $\mathbf{x}^0 = \mathbf{0}$ in Algorithm 1. Indeed, since $\mathbf{P}\mathbf{C}\mathbf{x}^0 = \mathbf{0}$ with $\mathbf{x}^0 = \mathbf{y}$ and with $\mathbf{x}^0 = \mathbf{0}$, it results that $\hat{\mathbf{r}}^0 = \mathbf{P}\mathbf{b}$ in the third line of Algorithm 1, leading to the same iterative process.

Second, the two first vectors associated with the largest singular values of $\mathbf{R}_s$ are computed using the set of 12 snapshots and are considered as deflation vectors, as described in Section 4. The two selected eigenvectors explain more than 99.9% of the variability of the 12 snapshots. The resulting DPCG method with a zero vector as initial solution vector converges in 1215 iterations (Fig. 4; Table 1). Based on these results, we conclude that the variability not captured by the two POD vectors slightly deteriorates the convergence in comparison to using all the 12 snapshots as deflation vectors, while

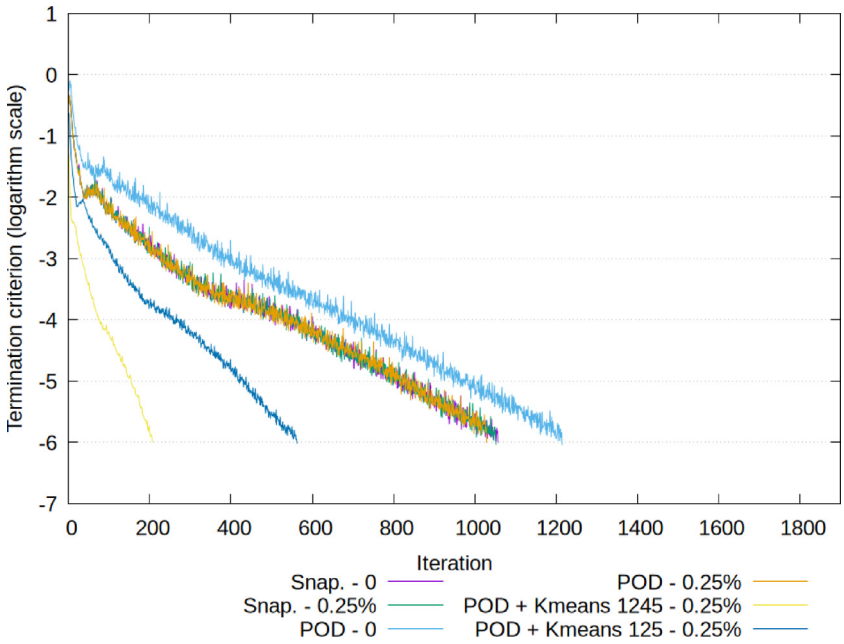

Fig. 4. Convergence plots for the DPCG method with the deflation vectors corresponding to 12 snapshots (Snap.) or to 2 POD vectors (POD), as well as in combination with deflation vectors defined with a subdomain deflation approach (Kmeans). The DPCG methods used either a zero initial solution vector (0) or a solution vector from a previous evaluation (0.25%).

they explain more than 99.9% of the variability of the 12 snapshots. However, using the solution vector of a previous evaluation with 0.25% less genotypes and phenotypes as initial solution vector in addition to the POD vectors, results in a number of iterations to reach convergence (i.e. 1029; Fig. 4; Table 1) similar to the amount of iterations needed when the 12 snapshots are used. This is interesting because this POD approach with a non-zero initial solution vector is computationally more efficient as only two deflation vectors are needed instead of 12 for a similar convergence rate.

Regarding the Ritz values, recycling snapshots, or using the POD vectors derived from them, as deflation vectors do not affect the largest Ritz values, since they remain similar to the largest Ritz values obtained with the PCG method. However, we observe that the smallest Ritz values slightly increase in comparison to the PCG method, when snapshots are recycled into deflation vectors. This is the opposite of the DPCG method using the subdomain deflation approach, as this approach results in a decrease of the largest Ritz values.

5.3.5. The DPCG method based on subdomain deflation and recycling information approaches

Based on the observations that the subdomain deflation approach results in a decrease of the largest Ritz values and that the recycling information approach results in a slight increase of the smallest Ritz values, we combine the 2 POD-based deflation vectors with the subdomain deflation vectors obtained from the K-means algorithm-based approach into a single deflation-subspace matrix $\mathbf{Z}_d$.

As observed in Fig. 4 and Table 1, the combination of both approaches results in the lowest numbers of iterations to reach convergence. Indeed, 207 and 563 iterations are needed when 1245 and 125 subdomains are defined for the SNP effects, respectively. The combined deflation-subspace matrices result in smallest Ritz values similar to the smallest Ritz values obtained with the POD approach, and in largest Ritz values similar to the largest Ritz values obtained with the K-means algorithm-based approach. These results show that both approaches to define the deflation-subspace matrix can be combined if both approaches result in annihilating different extreme eigenvalues of the preconditioned coefficient matrices.

5.4. Computational costs of the different PCG and DPCG approaches

In this section, we investigate first the computational costs of the PCG and DPCG approaches using small datasets, as in this study. Second, we discuss the computational costs of the PCG and DPCG approaches applied to large datasets (e.g., as in Vandenplas et al. [15]).

5.4.1. Small datasets

When small datasets include only a few tens of thousands of genotypes and SNPs, the matrices $\mathbf{M}$, $\mathbf{M}_k$, $\mathbf{C}$, $\mathbf{E}^{-1}$, and $\tilde{\mathbf{Z}} = \mathbf{C}\mathbf{Z}_d$, can be computed explicitly and stored into Random Access Memory (RAM). In this case, the differences of the

Table 2
Wall clock times for different steps of different PCG-based approaches applied on a ssSNPBLUP linear system.

Approach[a]	Deflation	# def. vect.	Init. sol.	Setup $\mathbf{M}$ (s)	Iterative process				
					Total[b] (s)	Iteration[c] (s)	$\mathbf{Mv} = \mathbf{r}$[d] (ms)	Setup $\mathbf{P}$ (s)	$\mathbf{Pv}$[e] (ms)
PCG	–	–	0	0.2	2621	1.50	6	–	–
PCG	–	–	0.25%	0.4	1869	1.66	6	–	–
PCG-B	–	–	0	10.5	406	1.54	102	–	–
PCG-B	–	–	0.25%	12.2	343	2.04	146	–	–
DPCG	Random	1246	0	<0.1	640	1.68	5	2.5	236
DPCG	Random	126	0	<0.1	1615	1.45	5	1.9	9
DPCG	Random	1246	0.25%	<0.1	457	1.84	5	3.3	239
DPCG	Random	126	0.25%	<0.1	1106	1.51	5	1.9	9
DPCG	K-means	1246	0	0.3	577	1.68	5	5.7	234
DPCG	K-means	126	0	0.3	1310	1.46	5	5.2	9
DPCG	K-means	1246	0.25%	<0.1	384	1.74	5	2.3	233
DPCG	K-means	126	0.25%	<0.1	871	1.49	5	2.0	9
DPCG	Snapshots	12	0	<0.1	1608	1.50	5	17.0	2
DPCG	Snapshots	12	0.25%	<0.1	1811	1.70	7	17.4	2
DPCG	POD	2	0	<0.1	1811	1.49	5	2.8	1
DPCG	POD	2	0.25%	<0.1	1535	1.49	5	3.2	1
DPCG	K-means + POD	1248	0.25%	<0.1	367	1.73	5	5.6	233
DPCG	K-means + POD	128	0.25%	<0.1	850	1.49	5	5.5	10

[a]PCG-B = PCG with a block Jacobi preconditioner; DPCG = deflated PCG.
[b]Wall clock time in seconds for the whole iterative solver, except the computation of the preconditioner $\mathbf{M}$.
[c]Average wall clock time in seconds for one iteration.
[d]Average wall clock time in milliseconds for solving the preconditioning system.
[e]Average wall clock time in milliseconds for one multiplication $\mathbf{Pv}$.

computational costs between different PCG-based methods are mainly due to differences between $\mathbf{M}$ and $\mathbf{M}_k$, and among the different sizes of $\mathbf{E}^{-1}$ and $\tilde{\mathbf{Z}}$.

First, in this study, the computation of $\mathbf{M}$ takes only a few milliseconds, and the computation of $\mathbf{M}_k$ takes between 10 and 12 s due to the Bunch–Kaufman factorization [38] (Section 5.3.2; Table 2). Similarly, the preconditioning system is solved within 7 ms on average with $\mathbf{M}$, and in around 146 ms with $\mathbf{M}_k$ (Table 2).

Second, the storage in RAM of $\mathbf{C}$ and $\tilde{\mathbf{Z}} = \mathbf{C}\mathbf{Z}_d$ allows the implementation of $\mathbf{PCp}^k$ in the DPCG method as follows [39]:

$$\mathbf{PCp}^k = \mathbf{y}^k - \left[\tilde{\mathbf{Z}}\left[\mathbf{E}^{-1}\left[\mathbf{Z}_d'\mathbf{y}^k\right]\right]\right] \tag{8}$$

where the brackets [.] indicate the order of the matrix–vector operations, $\mathbf{y}^k = \mathbf{Cp}^k$. Excluding the time required for computing $\mathbf{y}^k = \mathbf{Cp}^k$, performing Eq. (8) takes less than 10 ms with up to 126 deflation vectors, and around 240 ms with about 1250 deflation vectors (Table 2).

Finally, these different times result in slightly different wall clock times per iteration across the different PCG-based implementations. Thereby, one iteration takes on average between 1.5 and 2.0 s for all PCG-based methods (Table 2), and the efficiency of the different solvers is highly related to the wall clock times required for the computation of $\mathbf{M}$ (or $\mathbf{M}_k$), $\mathbf{E}^{-1}$ and $\tilde{\mathbf{Z}}$, and the number of iterations to reach convergence. Based on these criteria, a PCG method with a block Jacobi preconditioner or a DPCG method based on subdomain deflation and recycling information approaches can be advised for solving ssSNPBLUP systems with small datasets.

5.4.2. Large datasets

When large datasets include several hundreds of thousand genotypes and tens of thousand SNP markers, the computational strategies discussed in Section 5.4.1 might not possible. First, for the PCG methods, the computation of the block Jacobi preconditioner $\mathbf{M}_k$ may require large amounts of RAM for large n_s, as it involves a dense matrix $\left(\frac{1}{w}\mathbf{Z}'\mathbf{A}_{gg}^{-1}\mathbf{Z} + \frac{m}{1-w}\mathbf{I}\right) \in \mathbb{R}^{n_s \times n_s}$. Furthermore, its computation might be challenging in matrix-free solvers that use compressed forms of the dense matrix $\mathbf{Z}$ (e.g., [18]). Although the efficiency of $\mathbf{M}_k$ is attractive, its implementation for large datasets requires further research.

Second, the computational costs of the DPCG method compared to those of the PCG methods will also depend on how the product $\mathbf{PCp}^k$ in Algorithm 1 is implemented with large datasets. The implementation of the product $\mathbf{PCp}^k$ in this study (Eq. (8)) can be also used when the number of deflation vectors is limited because the matrices $\mathbf{Z}_d$, $\mathbf{E}^{-1}$, and $\tilde{\mathbf{Z}}$ can be stored in-memory and allow parallelized matrix–vector operations. This is the case for the DPCG method based on recycling information from previous systems.

However, this approach is not possible with the subdomain deflation approach because thousands of deflation vectors are required for real genomic evaluations, resulting in a matrix $\tilde{\mathbf{Z}}$ too large to be stored in-memory [13]. For such a case,

Vandenplas et al. [13] proposed to store the deflation subspace matrix $\mathbf{Z}_d$ in-memory as a sparse matrix and to compute the product $\mathbf{PCp}^k$ as follows [30]:

$$\mathbf{PCp}^k = \mathbf{y}^k - \left[\mathbf{C}\left[\mathbf{Z}_d\left[\mathbf{E}^{-1}\left[\mathbf{Z}_d'\mathbf{y}^k\right]\right]\right]\right] \tag{9}$$

This approach has the downside that the multiplication of $\mathbf{C}$ by a vector must be performed twice per iteration. Assuming that the costs of an iteration is dominated by the multiplication of $\mathbf{C}$ by a vector, the DPCG method that implements Eq. (9) is therefore more efficient than the PCG method if a reduction of the number of iterations by a factor greater than two is observed. Thereby, Eq. (8) should be preferred when the multiplication of $\tilde{\mathbf{Z}}$ by a vector is less expensive than the multiplication of $\mathbf{C}$ by a vector.

Regarding the computational costs of the DCPG method relying on the K-means algorithm-based approach, the additional costs of the K-means++ algorithm can be neglected. Indeed, in practice, genomic evaluations are performed routinely, and while the size of phenotypic, genomic, and pedigree information increase over time, we can assume that the correlation matrix associated with the genotype matrix remains similar across evaluations and across similar sets of animals [30,40]. Therefore, the K-means++ analysis of the genotype matrix could be performed only once for several genomic evaluations that use approximately the same pedigree and genomic information. Thereby, the computation costs of the DPCG method decreases linearly with the decrease of the number of DPCG iterations when using K-means algorithm-based approach, in comparison to the random assignment approach.

Regarding the computational costs of the DPCG method based on recycling information from previous systems, it is obvious that the DPCG method using the POD basis as deflation vectors require less operations per iteration than the DPCG method using snapshots. For a similar convergence rate, as observed in this study, the POD-based approach should therefore be preferred, because the extra work needed to compute the POD basis can be limited as detailed in Section 3 and by Diaz Cortes et al. [32].

6. Conclusions

In this study, we show that the convergence rate of the DPCG method applied to a ssSNPBLUP system can be improved by refining the definition of the deflation vectors based on the properties of the genomic information. This acceleration of the convergence is due to the annihilation of the largest eigenvalues of the ssSNPBLUP preconditioned coefficient matrix. Furthermore, the convergence rate of the DPCG method is accelerated by recycling solution vectors of previous ssSNPBLUP systems. The POD vectors obtained from solution vectors of previous systems can be used as deflation vectors to annihilate the smallest eigenvalues of the ssSNPBLUP preconditioned coefficient matrix. Finally, the DPCG method relying on both approaches for defining the deflation vectors and with the use of the solution vector of a previous ssSNPBLUP system, results in a reduction of the number of iterations up to 88% in comparison to the PCG method with a Jacobi preconditioner. Similar improvements are also observed with the PCG method using a block Jacobi preocondioner. Therefore, we recommend the PCG method with a block Jacobi preconditioner and the DPCG method that combines the subdomain deflation approach based on the properties of the genomic information and the POD-based approach recycling previous solution vectors for solving efficiently a ssSNPBLUP system of linear equations.

Data availability

The scripts to generate the datasets and the analyses are available at https://doi.org/10.4121/19153742 and at WUR Gitlab https://git.wur.nl/vande018/vandenplas_dpcg_2022.git.

Acknowledgements

This study was financially supported by the Dutch Ministry of Economic Affairs (TKI Agri & Food Project 16022) and the Breed4Food partners Cobb Europe (Colchester, Essex, United Kingdom), CRV (Arnhem, the Netherlands), Hendrix Genetics (Boxmeer, the Netherlands), and Topigs Norsvin (Helvoirt, the Netherlands). The use of the high-performance cluster was made possible by CAT-AgroFood (Shared Research Facilities Wageningen UR, Wageningen, the Netherlands).

References

[1] R.A. Mrode, Linear Models for the Prediction of Animal Breeding Values, second ed., CABI Publishing, Wallingford, UK, 2005.
[2] A. Legarra, O.F. Christensen, I. Aguilar, I. Misztal, Single step, a general approach for genomic selection, Livest. Sci. 166 (2014) 54–65, URL http://www.livestockscience.com/article/S1871141314002303/abstract.
[3] I. Aguilar, I. Misztal, D. Johnson, A. Legarra, S. Tsuruta, T. Lawlor, Hot topic: A unified approach to utilize phenotypic, full pedigree, and genomic information for genetic evaluation of Holstein final score, J. Dairy Sci. 93 (2) (2010) 743–752, URL http://linkinghub.elsevier.com/retrieve/pii/S0022030210715174.
[4] O.F. Christensen, M.S. Lund, Genomic prediction when some animals are not genotyped, Genet. Select. Evol. 42 (1) (2010) 2, URL http://www.gsejournal.org/content/42/1/2/abstract.
[5] I. Misztal, A. Legarra, I. Aguilar, Using recursion to compute the inverse of the genomic relationship matrix, J. Dairy Sci. 97 (6) (2014) 3943–3952, URL http://www.journalofdairyscience.org/article/S0022030214002240/abstract.

[6] E.A. Mäntysaari, R.D. Evans, I. Strandén, Efficient single-step genomic evaluation for a multibreed beef cattle population having many genotyped animals, J. Anim. Sci. 95 (11) (2017) 4728–4737, http://dx.doi.org/10.2527/jas2017.1912.

[7] Z. Liu, M. Goddard, F. Reinhardt, R. Reents, A single-step genomic model with direct estimation of marker effects, J. Dairy Sci. 97 (9) (2014) 5833–5850, URL http://www.journalofdairyscience.org/article/S0022030214004895/abstract.

[8] R.L. Fernando, J.C. Dekkers, D.J. Garrick, A class of Bayesian methods to combine large numbers of genotyped and non-genotyped animals for whole-genome analyses, Genet. Select. Evol. 46 (1) (2014) 50, URL http://www.gsejournal.org/content/46/1/50.

[9] R.L. Fernando, H. Cheng, B.L. Golden, D.J. Garrick, Computational strategies for alternative single-step Bayesian regression models with large numbers of genotyped and non-genotyped animals, Genet. Select. Evol. 48 (2016) 96, http://dx.doi.org/10.1186/s12711-016-0273-2.

[10] M. Taskinen, E.A. Mäntysaari, I. Strandén, Single-step SNP-BLUP with on-the-fly imputed genotypes and residual polygenic effects, Genet. Select. Evol. 49 (2017) 36, http://dx.doi.org/10.1186/s12711-017-0310-9.

[11] I. Strandén, M. Lidauer, Solving large mixed linear models using preconditioned conjugate gradient iteration, J. Dairy Sci. 82 (12) (1999) 2779–2787, URL http://www.journalofdairyscience.org/article/S0022030299755359/abstract.

[12] S. Tsuruta, I. Misztal, I. Stranden, Use of the preconditioned conjugate gradient algorithm as a generic solver for mixed-model equations in animal breeding applications, J. Anim. Sci. 79 (5) (2001) 1166–1172, URL http://www.journalofanimalscience.org/content/79/5/1166.short.

[13] J. Vandenplas, H. Eding, M.P.L. Calus, C. Vuik, Deflated preconditioned conjugate gradient method for solving single-step BLUP models efficiently, Genet. Select. Evol. 50 (1) (2018) 51, http://dx.doi.org/10.1186/s12711-018-0429-3.

[14] R. Nabben, C. Vuik, A comparison of deflation and the balancing preconditioner, SIAM J. Sci. Comput. 27 (5) (2006) 1742–1759, URL http://epubs.siam.org/doi/abs/10.1137/040608246.

[15] J. Vandenplas, M.P.L. Calus, H. Eding, C. Vuik, A second-level diagonal preconditioner for single-step SNPBLUP, Genet. Select. Evol. 51 (1) (2019) 30, http://dx.doi.org/10.1186/s12711-019-0472-8.

[16] N. Gengler, G. Nieuwhof, K. Konstantinov, M. Goddard, Alternative single-step type genomic prediction equations, in: Book of Abstracts of the 63rd Annual Meeting of the EAAP: 2016, Bratislava.

[17] B.-V. Nguyen, Two-Level Preconditioning Applied on the ssSnpblup Model, TU Delft, Delft, 2021, URL http://resolver.tudelft.nl/uuid:ea4f1db4-47fa-461c-a200-bf0f1b5ae7ad.

[18] J. Vandenplas, H. Eding, M. Bosmans, M.P.L. Calus, Computational strategies for the preconditioned conjugate gradient method applied to ssSNPBLUP, with an application to a multivariate maternal model, Genet. Select. Evol. 52 (1) (2020) 24, http://dx.doi.org/10.1186/s12711-020-00543-9.

[19] Y. Masuda, I. Misztal, A. Legarra, S. Tsuruta, D.a.L. Lourenco, B.O. Fragomeni, I. Aguilar, Technical note: Avoiding the direct inversion of the numerator relationship matrix for genotyped animals in single-step genomic best linear unbiased prediction solved with the preconditioned conjugate gradient, J. Anim. Sci. 95 (1) (2017) 49–52.

[20] K.V. Konstantinov, M.E. Goddard, Application of multivariate single-step SNP best linear unbiased predictor model and revised SNP list for genomic evaluation of dairy cattle in Australia, J. Dairy Sci. (2020) http://dx.doi.org/10.3168/jds.2020-18242, URL https://www.journalofdairyscience.org/article/S0022-0302(20)30506-3/abstract.

[21] Y. Saad, Iterative Methods for Sparse Linear Systems, second ed., in: Other Titles in Applied Mathematics, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003, http://dx.doi.org/10.1137/1.9780898718003.

[22] I. Strandén, M. Lidauer, Parallel computing applied to breeding value estimation in dairy cattle, J. Dairy Sci. 84 (1) (2001) 276–285, http://dx.doi.org/10.3168/jds.S0022-0302(01)74477-3, URL http://www.journalofdairyscience.org/article/S0022030201744773/abstract.

[23] M. Lidauer, I. Strandén, E.A. Mäntysaari, J. Pösö, A. Kettunen, Solving large test-day models by iteration on data and preconditioned conjugate gradient, J. Dairy Sci. 82 (12) (1999) 2788–2796, http://dx.doi.org/10.3168/jds.S0022-0302(99)75536-0.

[24] C. Vuik, R. Nabben, J. Tang, Deflation Acceleration for Domain Decomposition Preconditioners, 2006.

[25] Y. Saad, M. Yeung, J. Erhel, F. Guyomarc'h, A deflated version of the conjugate gradient algorithm, SIAM J. Sci. Comput. 21 (5) (2000) 1909–1926, http://dx.doi.org/10.1137/S1064829598339761, URL http://epubs.siam.org/doi/abs/10.1137/S1064829598339761.

[26] J.M. Tang, Two-level preconditioned conjugate gradient methods with applications to bubbly flow problems (Ph.D. thesis), TU Delft, Delft, 2008, URL http://repository.tudelft.nl/islandora/object/uuid:e8c5f63b-ee7d-4a59-90da-a8025f5f88b0?collection=research.

[27] C. Vuik, A. Segal, J.A. Meijerink, An efficient preconditioned CG method for the solution of a class of layered problems with extreme contrasts in the coefficients, J. Comput. Phys. 152 (1) (1999) 385–403, URL http://www.sciencedirect.com/science/article/pii/S0021999199962551.

[28] K. Burrage, J. Erhel, B. Pohl, A. Williams, A deflation technique for linear systems of equations, SIAM J. Sci. Comput. 19 (4) (1998) 1245–1260, http://dx.doi.org/10.1137/S1064827595294721, URL http://epubs.siam.org/doi/abs/10.1137/S1064827595294721.

[29] J. Frank, C. Vuik, On the construction of deflation-based preconditioners, SIAM J. Sci. Comput. 23 (2) (2001) 442–462, URL http://epubs.siam.org/doi/abs/10.1137/S1064827500373231.

[30] J. Vandenplas, M.P.L. Calus, G. Gorjanc, Genomic prediction using individual-level data and summary statistics from multiple populations, Genetics 210 (1) (2018) 53–69, http://dx.doi.org/10.1534/genetics.118.301109, URL http://www.genetics.org/content/210/1/53.

[31] D. Arthur, S. Vassilvitskii, k-means++: The Advantages of Careful Seeding, Stanford, 2006, URL http://ilpubs.stanford.edu:8090/778/.

[32] G.B. Diaz Cortes, C. Vuik, J.D. Jansen, On POD-based deflation vectors for DPCG applied to porous media problems, J. Comput. Appl. Math. 330 (2018) 193–213, http://dx.doi.org/10.1016/j.cam.2017.06.032, URL http://www.sciencedirect.com/science/article/pii/S0377042717303400.

[33] M. Sargolzaei, F.S. Schenkel, QMSim: a large-scale genome simulator for livestock, Bioinformatics 25 (5) (2009) 680–681, URL http://bioinformatics.oxfordjournals.org/content/25/5/680.

[34] H.L. Bradford, Y. Masuda, J.B. Cole, I. Misztal, P.M. VanRaden, Modeling pedigree accuracy and uncertain parentage in single-step genomic evaluations of simulated and US Holstein datasets, J. Dairy Sci. (2019) http://dx.doi.org/10.3168/jds.2018-15419, URL http://www.sciencedirect.com/science/article/pii/S0022030219300426.

[35] J. Bezanson, A. Edelman, S. Karpinski, V.B. Shah, Julia: A fresh approach to numerical computing, SIAM Rev. 59 (1) (2017) 65–98, http://dx.doi.org/10.1137/141000671.

[36] E.F. Kaasschieter, A practical termination criterion for the conjugate gradient method, BIT 28 (2) (1988) 308–322, http://dx.doi.org/10.1007/BF01934094, URL https://link.springer.com/article/10.1007/BF01934094.

[37] C. Paige, M. Saunders, Solution of sparse indefinite systems of linear equations, SIAM J. Numer. Anal. 12 (4) (1975) 617–629, URL http://epubs.siam.org/doi/abs/10.1137/0712047.

[38] J.R. Bunch, L. Kaufman, Some stable methods for calculating inertia and solving symmetric linear systems, Math. Comp. 31 (137) (1977) 163–179, http://dx.doi.org/10.1090/S0025-5718-1977-0428694-0, URL https://www.ams.org/mcom/1977-31-137/S0025-5718-1977-0428694-0/.

[39] T.B. Jönsthövel, M.B.v. Gijzen, S. MacLachlan, C. Vuik, A. Scarpas, Comparison of the deflated preconditioned conjugate gradient method and algebraic multigrid for composite materials, Comput. Mech. 50 (3) (2012) 321–333, http://dx.doi.org/10.1007/s00466-011-0661-y, URL https://link.springer.com/article/10.1007/s00466-011-0661-y.

[40] R.M. Maier, Z. Zhu, S.H. Lee, M. Trzaskowski, D.M. Ruderfer, E.A. Stahl, S. Ripke, N.R. Wray, J. Yang, P.M. Visscher, M.R. Robinson, Improving genetic prediction by leveraging genetic correlations among human diseases and traits, Nature Commun. 9 (1) (2018) 989, URL https://www.nature.com/articles/s41467-017-02769-6.