A Collection of Numerical Recipes Useful for Building Scalable Psychometric Applications

Harold Doran
hdoran@humrro.org
Human Resources Research Organization
Working Technical Paper

April 9, 2022

Abstract

This paper is concerned with a subset of numerically stable and scalable algorithms that can be useful to support computationally complex psychometric models in the era of machine learning and massive data. The subset selected here is a core set of numerical methods that should be familiar to computational psychometricians and considers whitening transforms for dealing with correlated data, computational concepts for linear models, multivariable integration, and optimization techniques.

Keywords: computational psychometrics; gaussian quadrature; item response theory; linear models; numerical analysis; psychometric data scientist

Psychometrics is increasingly becoming a multidisciplinary field more directly blended with modern computing. Machine and deep learning models are becoming central psychometric methods and present some exciting ways to evaluate psychological constructs. These new opportunities also introduce new challenges. Massive data are now typical and advances in machine learning stress existing computational infrastructures to a significant degree. As a result, quantitative disciplines such as psychometrics must confront the changing dynamics of large data and computationally demanding psychometric methods.

The National Research Council's (NRC) report on massive data addressed how data science is changing and highlighted that extending scientific reach will require developing scalable computational infrastructure with statisticians explicitly focused on scalability, algorithms, numerical linear algebra, optimization, and adapt to new types of challenging computational demands (National Research Council, 2013). Many scientists working to build scalable systems often tackle the idea by considering different software (e.g., using a compiled language rather than an interpreted language) or engaging larger, cloud-based infrastructures or distributed computing concepts. These are very

useful conversations to engage in; however, the central focus offered here is to first consider the smallest and most manageable unit—the numerical methods used in the underlying code to build models. These are the core at which scientific ideas are implemented and computational psychometricians require a portfolio of efficient, scalable techniques at the ready so that new ideas can unfold. Then, good computational methods can more readily take advantage of larger computing environments than expecting poor methods to be remedied by more powerful computing.

In fact, the disciplines of machine learning and scientific computing are now effectively a blended discipline (Innes et al., 2019) with advancements resting on the ideas made available via numerical analysis. It's useful therefore to assess the preparedness of quantitative disciplines such as psychometrics with respect to numerical analysis to evaluate our current position relative to the skill set imagined by the NRC report. A few examples suggest a gap in preparedness. Bates (2004a) noted that the algebraic textbook solution for the least squares model is commonly used within R code and provided a tutorial demonstrating a numerically stable and fast implementation. Many popular online forums supporting code development used by data scientists, such as Medium, often convey the numeric solution to the least squares problem in the same form as the least squares notational expression and unnecessarily use matrix inversion (Koseoglu, 2018). Genz and Kass (1997) report that statisticians often ignore superior methods for high-dimensional integration and demonstrated how a deterministic approach can be tractable and more efficient than stochastic methods often used in Bayesian analysis. Antal and Oranje (2007) report that suboptimal numerical methods are commonly found in psychometric software applications for high-dimensional integration finding that rectangular rules are common even in the National Assessment of Educational Progress (Allen, Donoghue, & Schoeps, 2001) and commercially available psychometric software (Muraki & Bock, 1999). Andersson and Xin (2021) describe the rectangular quadrature approach as being the most common numerical method used with integrals up to three dimensions in IRT applications.

Informally, a common theme observed is that code is often written in ways that mirrors the notational representation of a problem. Notational representations are of course valuable for developing a conceptual understanding, but notational expressions should not be interpreted as the same way to build a high performance computational implementation. In fact, this practice often leads to expensive and unsophisticated numerical methods and building applications this way tends to limit innovation by preventing new applications from scaling with the new types of real world data and computational demands.

Notably, there are interesting conversations among computational psychometricians regarding the challenges in building scalable applications. Cai, Yang, and Hansen (2011) discuss the specific computational challenges when faced with high-dimensional integration problems as do the developers of the R package Dire (Doran, Bailey, Buehler, & joo Lee, 2021) for estimating the marginal likelihood regression (Mislevy, 1984) with useful alternatives such as the Metroplis-Hastings Robbins-Monro (MH-RM) method (Cai, 2010) or high-order Laplace approximations proposed (Andersson & Xin, 2021). Rijmen (2009) demonstrated how dimensionality reduction techniques can be applied to multidimensional integrals found in IRT applications to significantly reduce computational complexity. Rosseel (2021) provides an impressive example of psychometric computing that translates a complex problem into a feasible computational solution making excellent use of a standard inversion lemma and taking advantage of special matrix structures to a yield a simpler computational solution. More broadly, some advanced computational environments are emerging making use of many new scalable computational options, such as Julia (Bezanson,

Edelman, Karpinski, & Shah, 2017), TensorFlow (Abadi et al., 2015), and PyTorch (Paszke et al., 2019).

These progressive conversations related to algorithms and scalability have only one minor downside—they are commonly centered on a specific application or problem and so many computational ideas are spread across disparate resources. In contrast, the work presented here is intentionally problem agnostic and offers an organized survey of tested numerical methods with the intent that psychometricians will have a consolidated resource with templates to springboard new, computationally demanding psychometric ideas. The recipes here are a subset of many possible ways to compute, but as observed in other places, new algorithms such as MH-RM (Cai, 2010) or importance Gauss-Hermite (Elvira, Martino, & Closas, 2021) are extensions of these first principles. So, appreciating this subset can be a toolkit upon which other novel approaches are extended.

There are three goals of this paper including 1) to methodically step through a collection of useful numerical recipes 2) explicitly link and build connections between different approaches and 3) illustrate their potential in some common psychometric situations. Some of the methods presented here on their own are simple, but in combination they form a portfolio of powerful techniques that can be used to build and scale challenging psychometric ideas. Some unique combinations of the ideas are introduced in this paper, such as combining stochastic gradient descent with a parallel expectation maximization problem and the Woodbury identity is combined with an eigendecomposition approach useful for matrix inversions in iterative calculations.

The hope is to create an accessible resource and so this work assumes a tutorial style and is oriented more towards application and does not set out to formally prove theorems or engage in deriving estimators and proving their properties other than one simple proof in the appendix. The recipes are introduced in the framework of familiar and simple problems and then each idea is advanced sequentially to more complex scenarios. Four general topics are covered including whitening transforms for correlated data, recipes for linear models, integration, and optimization all with an emphasis on numerical stability and scalability. This paper is organized to present each topic in sequence and then illustrate their potential applications with three computational examples not traditionally found in standard software.

There are three concepts emphasized and demonstrated in different ways throughout this paper. First, always use a numerically stable algorithm. Second, where possible, precompute and recycle expensive components in iterative algorithms. Last, rewrite problems to identify simplified, reduceable structures.

Notation and Terminology

Standard mathematical notation is used throughout and intended interpretations are described within each relevant section. An important distinction in terminology is offered using "algebraic solution" to mean the way a solution to a problem is written to convey a concept. In contrast, a "computational solution" is a numerical recipe that provides a stable way to implement a solution in software to estimate parameters that may not resemble the algebraic solution. The distinction between the two is a non-trivial point and it's important to be clear that code should be not necessarily be transcribed to mirror mathematical notations.

Defining Scalable Psychometric Applications

To remove ambiguity, "scale" in this work refers to the ability of an algorithm (i.e., the computational implementation) and the system making use of the algorithm (i.e., the software application) to handle big inputs and grow in relationship to the size of the problem (Teng, 2016). The algorithm is how the idea is implemented and the application is how that idea becomes available to others.

There are three primary ways input sizes tend to grow. These include 1) the magnitude of the data used for analysis, 2) the complexity of the computational problem, and 3) the large number of simultaneous users accessing a common system. A scalable application is one that can flexibly handle all three components efficiently. Computational complexity is considered as an input here because many complex problems require large inputs. For instance, high-dimensional integrals require quadrature points that increase exponentially with the number of dimensions.

Traditional psychometric techniques, such as item calibrations in large populations, are commonly managed via sampling as a data reduction method. This has been convenient for decades with simple, low-dimensional psychometric models that require relatively small computations with software installed on a local machine. In essence, this statement captures how scale is managed in traditional psychometrics—make big data small, use simple computational methods, and rely on single user applications installed on a local machine. However, the future of psychometrics will bring new models using massive data in ways that will break the traditional psychometric paradigm and they will require the richness associated massive data, the benefits of complex psychometric models in high-dimensional space, and the modern nature of computing in a cloud-based infrastructure.

Introductory Concepts for Dealing with Correlated Data

Whitening Transformations for Correlated Variables

A concept first introduced and used throughout almost all subsequent recipes is referred to as the whitening transformation. Whitening is a process of transforming a collection of correlated vectors into a collection of uncorrelated vectors with the aid of a suitable decomposition (Kessy, Lewin, & Strimmer, 2018). The process is critical to the solution of linear systems and for fast quadrature routines and is generally helpful for reducing complex problems into more computationally feasible solutions.

The concept is introduced as follows. Let $\boldsymbol{Y} \in \mathbb{R}^{n \times p}$ be a matrix of correlated variables from a p-dimensional multivariate normal, $\boldsymbol{Y} \sim \mathcal{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\mu} \in \mathbb{R}^p$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ and is positive definite. Now find a decomposition that satisfies $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}'$ where $\boldsymbol{L}$ is lower triangular known as the Cholesky factor, and $\boldsymbol{\Sigma}^{-1} = \boldsymbol{L}'^{-1}\boldsymbol{L}^{-1}$. This provides that the transformation $\boldsymbol{X} = \boldsymbol{Y}\boldsymbol{L}'^{-1}$ can be used so that the variables in the matrix $\boldsymbol{X}$ are "whitened" or "de-correlated". Notice this can also be used to create correlated vectors when the operation is reversed, $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{L}'$, which may be useful for data generation, simulation, or applications such as high-dimensional integration.

There are many useful decompositions in numerical analysis (Searle, 1982; Zhang, 1999) and there are reasons to choose others depending on the computational objective. However, the Cholesky is useful for decorrelation and will be a focus in this work as it relates to the computational objectives described here and some work has demonstrated its potential advantage over

others, such as the QR when used for solving normal equations (Lira, Iyer, Trindade, & Howle, 2016). Some applications have fully adopted a sparse Cholesky decomposition (Davis & Hager, 2005) for linear mixed models (Bates, 2004b) with tremendous success.

Multivariate Densities of Whitened Data

The multivariate normal density function has an important role in psychometric practice for dealing with correlated Gaussian distributed random variables. It is possible to reduce this from a complex problem and show how the multivariate density can be computed as a product of transformed univariate densities. This change of variable transformation has broader applications for psychometric computing in maximization methods or dealing with high-dimensional integration problems. The general form of the multivariate normal with random vector $\boldsymbol{y} \in \mathbb{R}^p$, mean vector $\boldsymbol{\mu} \in \mathbb{R}^p$, and positive definite covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}' \in \mathbb{R}^{p \times p}$ is

$$f_y(y_1, \ldots, y_p) = \frac{\exp(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^p |\boldsymbol{\Sigma}|}} \tag{1}$$

If the mean vector is $\boldsymbol{\mu} = \mathbf{0}$ and $\boldsymbol{\Sigma} = diag(1, \ldots, 1)$, then the density of the multivariate normal is equal to the product of the marginal densities. Let $\varphi(\cdot)$ denote the standard normal probability density function (p.d.f.), then

$$f_y(y_1, \ldots, y_p) = \prod_{i=1}^{p} \varphi(y_i) \tag{2}$$

where (1) and (2) are equivalent only under the special case of uncorrelated variables with unit variance. With correlated variables, rewrite (1) as

$$f_y(y_1, \ldots, y_p) = \frac{\exp(-\frac{1}{2}(\boldsymbol{y} - \boldsymbol{\mu})'\boldsymbol{L}'^{-1}\boldsymbol{L}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^p |\boldsymbol{L}\boldsymbol{L}'|}} \tag{3}$$

$$= \frac{\exp(-\frac{1}{2}\boldsymbol{y}'^{*}\boldsymbol{y}^{*})}{\sqrt{(2\pi)^p |\boldsymbol{L}^2|}} \tag{4}$$

$$= |\boldsymbol{L}|^{-1} \left(\prod_{i=1}^{p} \varphi(y_i^*) \right) \tag{5}$$

where $\boldsymbol{y}^* = \boldsymbol{L}^{-1}(\boldsymbol{y} - \boldsymbol{\mu})$ with elements $\boldsymbol{y}^* = \{y_1^*, \ldots, y_p^*\}'$. This notation implies inversion of $\boldsymbol{L}$. However, using forward substitution with the triangular system $\boldsymbol{L}\boldsymbol{y}^* = (\boldsymbol{y} - \boldsymbol{\mu})$ to solve for $\boldsymbol{y}^*$ is a numerically stable implementation. This transform expresses (5) into the form of (2), showing how the multivariate density can be computed as a product of the marginal densities after whitening. The univariate version of (5) is an example of a reduceable structure that simplifies the algebraic representation of (1) by avoiding matrix inverses and a key benefit of triangular matrices is that the determinant is very easily evaluated as the product of its diagonal elements.

Computational Concepts for Linear Systems

This section explores efficient computational recipes for linear models to create simple, reduceable structures to build stable and scalable psychometric algorithms. It's best to begin with a simple,

common problem—ordinary least squares (OLS), that will serve as a foundation and subsequently show how to frame more complex models with the same OLS solution. While the examples are provided within the context of linear regressions, the same concepts are used widely across psychometric practice. For example, these same methods are often used in the expectation-maximization (EM) algorithm during the M-step, in natural language processing problems, latent semantic analysis, solutions to factor analysis problems, in highly parameterized item calibrations problems using the Newton-Raphson algorithm, and within iteratively reweighted least solutions for generalized linear models. These are just a subset of the possible applications.

Ordinary Least Squares as a Unifying Framework

The standard representation of the linear least squares problem is via the following normal equations (McCulloch & Searle, 2001)

$$\boldsymbol{X'X\beta} = \boldsymbol{X'y} \tag{6}$$

where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a design matrix of full column rank, $\boldsymbol{\beta} \in \mathbb{R}^p$ is a vector of unknown parameters to be estimated, and $\boldsymbol{y} \in \mathbb{R}^n$ is a vector of observed outcomes. Provisionally assume in the following examples $p \ll n$, however many machine learning (ML) applications are trained on data when $n < p$, a problem referred to as high-dimensional regression. The algebraic representation in standard notation is shown as having an invertible $\boldsymbol{X'X}$ with a solution written as

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X'X})^{-1}\boldsymbol{X'y}. \tag{7}$$

It is common to observe in various texts and online forums strong statements around the representation shown in Equation (7) with claims insinuating that this is also its computational implementation. In fact, this is a poor implementation and it is unnecessary to invert $\boldsymbol{X'X}$. Instead, a more stable approach is to find the decomposition, $\boldsymbol{X'X} = \boldsymbol{LL'}$ where $\boldsymbol{L}$ is the Cholesky factor, and then rewrite (6) as $\boldsymbol{LL'\beta} = \boldsymbol{X'y}$. This expression provides a simple solution via two triangular systems as

$$\begin{aligned} \boldsymbol{Lz} &= \boldsymbol{X'y}, \text{ forward substitution for } \boldsymbol{z} \\ \boldsymbol{L'\beta} &= \boldsymbol{z}, \text{ backward substitution for } \boldsymbol{\beta}. \end{aligned} \tag{8}$$

The first triangular system solves for $\boldsymbol{z}$ and the second transforms the $p \times 1$ vector $\boldsymbol{z}$ by the Cholesky factor to yield the estimates for the $p \times 1$ vector $\boldsymbol{\beta}$. Obtaining least squares estimates is very simple and (8) shows estimates for $\boldsymbol{\beta}$ are obtained as simple expressions of $\boldsymbol{z}$ and $\boldsymbol{\beta}$ and require no matrix inversions. However, it serves as a baseline and a general unifying framework for computing linear systems as subsequently shown when more complex models can be rewritten into the form of (6). In such cases, other models can be shown to inherit the solution shown in (8) even when they appear to be more complex, hence it unifies approaches to computing linear systems.

Generalized Least Squares in OLS Form

The generalized least squares (GLS) problem is an incrementally more advanced model than its OLS counterpart. GLS is commonly needed in psychometrics when the assumptions of the Gauss-

Markov theorem do not tend to hold, namely the assumptions of homoscedastic variance and uncorrelated errors. In fact, the latter assumption of uncorrelated errors is almost always violated in clustered settings (Hedges & Hedberg, 2007) and GLS offers one pathway for dealing with correlated observations. The standard textbook normal equations for the GLS model is

$$\boldsymbol{X'\Sigma^{-1}X\beta = X'\Sigma^{-1}y} \tag{9}$$

where $\boldsymbol{\Sigma = LL'} \in \mathbb{R}^{n \times n}$ is a square, symmetric, positive definite covariance matrix and all other matrices are as previously noted in the least squares section. The algebraic solution for the GLS approach is commonly shown to involve matrix inversions as follows

$$\boldsymbol{\widehat{\beta} = (X'\Sigma^{-1}X)^{-1}X'\Sigma^{-1}y}. \tag{10}$$

The algebraic representation in (10) implies inverting the matrices $\boldsymbol{\Sigma}$ and $\boldsymbol{(X'\Sigma^{-1}X)}$ which can be entirely avoided. The dimensions of $\boldsymbol{\Sigma}$ may very large as it is equal to the number of rows in $\boldsymbol{X}$ and the GLS problem often involves an iterative solution with elements of $\boldsymbol{\Sigma}$ changing over the iterations, so inversions would be costly if not impossible. The concept of whitening can again be used to rewrite the normal equations in (9) such that the data in $\boldsymbol{X}$ are "de-correlated" as

$$\boldsymbol{X'^{*}X^{*}\beta = X'^{*}y^{*}} \tag{11}$$

where $\boldsymbol{X^{*} = L^{-1}X}$ and $\boldsymbol{y^{*} = L^{-1}y}$. It is again pointed out here this algebraic notation implies inversion of $\boldsymbol{L}$, but implementation forward solves for $\boldsymbol{y^{*}}$ in $\boldsymbol{Ly^{*} = y}$ and for $\boldsymbol{X^{*}}$ in $\boldsymbol{LX^{*} = X}$. This whitening transformation yields (11), reframing the GLS problem to assume the OLS form of (6) so that it can inherit the same solution shown in (8).

If (11) were used in an iterative process and elements of $\boldsymbol{\Sigma}$ were updated at each step, then recomputing its Cholesky factor repeatedly would layer additional expense. One consideration for managing computational overhead in iterative scenarios is demonstrated in the section on linear mixed models where a very large matrix is repeatedly updated in an iterative algorithm and finding its inverse at each iteration is costly.

Weighted Least Squares in OLS Form

Weighted least squares (WLS) is a special case of GLS when the covariance matrix $\boldsymbol{\Sigma}$ is diagonal. Like GLS, this is used in scenarios where the assumption of homoscedastic variance does not tend to hold, although WLS assumes uncorrelated errors across observations. In this case, it is entirely unnecessary to create and store the entire matrix $\boldsymbol{\Sigma}$ and instead only its diagonal elements, v_{jj}, are needed. Instead, there is a special matrix property providing that $\boldsymbol{X'\Sigma^{-1}X = X'X^{*}}$ where $\boldsymbol{X^{*}}$ is arrived at via the elementwise calculation of $1/v_{jj}$ over each column of the matrix $\boldsymbol{X}$. Performing this elementwise calculation to the matrix $\boldsymbol{X}$ on the left and right hand side of (9) when $\boldsymbol{\Sigma}$ is diagonal yields $\boldsymbol{X'X^{*}\beta = X'^{*}y}$, which is now written in the form of (6) and inherits the same OLS solution as (8). It is useful to point out that the Cholesky factor of a positive definite diagonal matrix is nothing more than the square root of the diagonal values of the original matrix. For example, $\boldsymbol{L^{-1}} = diag(\sqrt{1/v_{11}}, \sqrt{1/v_{22}}, \ldots, \sqrt{1/v_{jj}})$, providing a way to "skip ahead" and find an inverse and Cholesky factor at one time which could then be used in the same way as the solution in (11).

Stochastic Gradient Descent for Linear Systems

The previously discussed methods provide a coherent and numerically stable way to compute linear models in the form of (6). However, when the dimensions of the model matrix $\boldsymbol{X}$ become extremely large, then computing by (8) may prove challenging. It's difficult to offer a precise definition of "extremely large". One theoretical example is a quasi-matrix defined as some matrix having one infinite dimension (Shustin & Avron, 2021). Perhaps more concretely, we can imagine the vector space needed in natural language processing applications to represent words in examinee essay responses as one common example leading to a very large $\boldsymbol{X}$. We face at least two challenges in scenarios where $\boldsymbol{X}$ with dimensions $n \times p$ becomes large. First, it's simply difficult to create and store that model matrix in memory and if it can be created and stored, there is less overhead memory left for computing. Second, the computational complexity of matrix calculations does not always increase linearly. For example, the complexity of $\boldsymbol{X'X}$ occurs in $\mathcal{O}(np^2)$, thus becoming prohibitive with increasing dimensions.

Stochastic gradient descent (SGD) is a powerful approach for building highly scalable applications that reduces the computational burden associated with large data by simplifying it to an iterative solution passing over the gradient one sample at a time (Cizek & Cizkova, 2004; Shamir & Zhang, 2013; Tran, Toulis, & Airoldi, 2015). While SGD is a general purpose optimization technique useful for minimization of any differentiable objective (loss) function, $\mathcal{J}(\boldsymbol{\theta})$, it is presented within the section on linear models as it provides a fast and clear pathway for many common psychometric solutions, such as with ML applications. It is often the core algorithm used to construct neural networks in deep learning and is proving to be an extremely useful computational technique. Other general purpose optimization methods are discussed in a later section of this paper.

Formally, let $\nabla\mathcal{J}_i(\boldsymbol{\theta})$ denote the gradient vector of the objective function with respect to the parameters to be minimized for an ith sample in the data. SGD performs the optimization by drawing samples from the observed data and repeatedly updating the gradient and the parameters at each iteration as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \alpha\nabla\mathcal{J}_i(\boldsymbol{\theta}) \tag{12}$$

where $\alpha > 0$ is a learning rate parameter, $\boldsymbol{\theta}_t$ is the provisional value of the parameters at iteration t, and $\nabla\mathcal{J}_i(\boldsymbol{\theta})$ is the value of the gradient computed using sample i.

With respect to models assuming the form of (6), the gradient is $\nabla\mathcal{J}(\boldsymbol{\theta}) = 2n^{-1}(\boldsymbol{X'X\theta} - \boldsymbol{X'y})$. However, very large dimensions in $\boldsymbol{X}$ make general optimization difficult with either general gradient descent or using the traditional closed form solution in (8). Instead, the iterative process outlined in Algorithm (1) allows for a large problem to be reduced simply.

Algorithm 1 Stochastic Gradient Descent

Input: Choose initial values for $\boldsymbol{\theta}$, $\nabla\mathcal{J}_i(\boldsymbol{\theta})$, and select a value for α.

1: Set $\boldsymbol{\theta}_{t+1} := \boldsymbol{\theta}_t - \alpha\nabla\mathcal{J}_i(\boldsymbol{\theta})$ at iteration t.
2: Sample $\boldsymbol{X}_i$, in $\boldsymbol{X}$ and its corresponding value, $\boldsymbol{y}_i$.
3: Compute $\nabla\mathcal{J}_i(\boldsymbol{\theta})$ using the tuple $\{\boldsymbol{X}_i, \boldsymbol{y}_i\}$.
4: Iterate between steps (1) and (3) until criterion for a stopping rule has been satisfied.

When the sample size is equal to 1, then each iteration chooses one random row of the model matrix $\boldsymbol{X}$ and its corresponding value in the vector $\boldsymbol{y}$ and this process is termed stochastic gradient

descent. When $1 < i < n$, then each iteration selects i random rows in the model matrix $\boldsymbol{X}$ and the corresponding values in $\boldsymbol{y}$ and the same iterative process can be used in small batches termed mini-batch gradient descent. When $i = n$ with n being the total rows in $\boldsymbol{X}$, the entire training data set is used and this is termed batch gradient descent.

SGD, in contrast to traditional least squares algorithms, works only with samples of data and not with the full model matrix $\boldsymbol{X}$ at any point. This hints towards a different way to manage the amount of data needed in storage and the concept of streaming in only subsets of data needed for computing instead of requiring the complete data for memory management can be used with SGD (Tran et al., 2015). Consequently, SGD not only uses smaller computations, it also provides a means for using smaller subsets of data as it computes and the combination of the two is pivotal for computing with modern data that may contain many millions of observations.

A concept only noted in passing is that α is a learning rate *hyperparameter*, a term used differently in machine learning than by Bayesian statisticians meaning a tuning parameter used in training and selecting values for this scalar are not fully explored here. Pragmatically speaking, choosing values for the learning rate parameter is a harder problem that it may seem at the surface and there is a good deal of research to further explore. A "large" value for learning is generally desired for initial evaluations of the gradient allowing the algorithm to take large steps towards the optimal value. However, using that same large value when approaching the optimum may cause for the algorithm to jump over the optimal value and not converge to the expected result. Hence, the learning rate *schedule* is a concept that adjusts the learning rate to smaller values as the algorithm proceeds. In addition, the example here assumes $p \ll n$, in which case the loss function associated with standard linear regression can be used. In fact, many ML applications involve scenarios where $n < p$, in which case other approaches such as Ridge or Lasso regressions may prove useful and would simply adopt the general SGD framework described in Algorithm (1).

As an aside, iterative algorithms need a stopping rule, often referred to as convergence criteria. In many respects, it's an arbitrary rule constructed for a given condition and there are multiple ways to achieve this outcome. For example, one might compute changes in the parameter estimates from iteration t to $t + 1$ and take the difference between them. Stopping might occur if no parameter differs by more than ϵ, a term set to be very small indicating the parameters are not changing enough from iteration to iteration any longer or one might stop when differences in the log-likelihood differ by less than ϵ over iterations. In the SGD scenario, one might stop after reaching a fixed number of iterations or when the sum of the gradients squared is less than ϵ. No particular rule can be highly endorsed, although some scenarios are susceptible to being stuck in a saddle point.

Linear Mixed Models with Henderson's Method

The linear mixed model is generally written with the form $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{e}$ (Laird & Ware, 1982) and is often presented as having the normal equations (McLean, Sanders, & Stroup, 1991)

$$\begin{bmatrix} \boldsymbol{X'\Omega^{-1}X} & \boldsymbol{X'\Omega^{-1}Z} \\ \boldsymbol{Z'\Omega^{-1}X} & \boldsymbol{Z'\Omega^{-1}Z + G^{-1}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta} \\ \boldsymbol{u} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X'\Omega^{-1}y} \\ \boldsymbol{Z\Omega^{-1}y} \end{bmatrix} \tag{13}$$

where the following new notation is introduced $\boldsymbol{Z} = [\boldsymbol{Z}_1, \boldsymbol{Z}_2, \ldots, \boldsymbol{Z}_Q] \in \mathbb{R}^{n \times Q}$ is a model matrix for the random effects, $\boldsymbol{u} = [\boldsymbol{u}_1', \boldsymbol{u}_2', \ldots, \boldsymbol{u}_Q']' \in \mathbb{R}^Q$ is the vector of random effects, and $\boldsymbol{e} \in \mathbb{R}^n$ is

the residual error term. The matrices $\boldsymbol{\Omega} \in \mathbb{R}^{n \times n}$ and $\boldsymbol{G} \in \mathbb{R}^{Q \times Q}$ can have many forms depending on the structure of the model and are developed below for one variant of the linear mixed model. The term "level" is used here to mean a level of random variation. That is, a two-level model has two levels of random variation, a three-level model has three levels of random variation and so on. This term has a rather intuitive meaning when discussing models that are purely nested and a somewhat more challenging interpretation when considering models with fully or partially crossed random effects.

Algorithm 2 Henderson Mixed Model Sketch

Input: Create starting values for σ^2_ϵ and $\sigma^2_q \; \forall \; q$.

1: Construct $\boldsymbol{\Omega} = \sigma^2_\epsilon \boldsymbol{I}_n$ and $\boldsymbol{G} = diag(\{\sigma^2_1, \ldots, \sigma^2_1\}, \{\sigma^2_2, \ldots, \sigma^2_2\}, \ldots, \{\sigma^2_Q, \ldots, \sigma^2_Q\})$. The length of the diagonal element for block q is equal to the number of columns in its corresponding matrix, $\boldsymbol{Z}_q$.
2: Solve the linear system for $\boldsymbol{\beta}$ and $\boldsymbol{u}$.
3: Update the values of the variances of the random effects including σ^2_ϵ and σ^2_q.
4: Iterate between steps (1) and (3) until criterion for a stopping rule has been satisfied.

Given the representation of (13) there is a temptation to assume a naive solution to the linear system by inversion of the leftmost matrix. This is in fact how the SAS software documents the solution to the mixed model in its technical manual (SAS Documentation 14.2, n.d.). This may be perhaps just the algebraic representation and not actually the computational implementation. Instead, the computational solution can be written to resemble the solutions to linear systems previously described and a sketch of the iterative process is provided in Algorithm (2). Begin by writing

$$\boldsymbol{LL}' = \begin{bmatrix} \boldsymbol{X}'\boldsymbol{\Omega}^{-1}\boldsymbol{X} & \boldsymbol{X}'\boldsymbol{\Omega}^{-1}\boldsymbol{Z} \\ \boldsymbol{Z}'\boldsymbol{\Omega}^{-1}\boldsymbol{X} & \boldsymbol{Z}'\boldsymbol{\Omega}^{-1}\boldsymbol{Z} + \boldsymbol{G}^{-1} \end{bmatrix}, \quad \boldsymbol{y}^* = \begin{bmatrix} \boldsymbol{X}'\boldsymbol{\Omega}^{-1}\boldsymbol{y} \\ \boldsymbol{Z}\boldsymbol{\Omega}^{-1}\boldsymbol{y} \end{bmatrix} \tag{14}$$

where elementwise calculation can be used for the diagonal matrices (e.g., $\boldsymbol{\Omega}^{-1}\boldsymbol{X}$) as presented in the WLS section and $\boldsymbol{\Theta}' = [\boldsymbol{\beta}' \; \boldsymbol{u}']$. The model can now be written as $\boldsymbol{LL}'\boldsymbol{\Theta}' = \boldsymbol{y}^*$ sharing the representation of the OLS form and also inherits the solution using (8) through the two triangular systems, $\boldsymbol{Lz} = \boldsymbol{y}^*$ for $\boldsymbol{z}$ and then $\boldsymbol{L}'\boldsymbol{\Theta} = \boldsymbol{z}$ for $\boldsymbol{\Theta}$. Using the Cholesky in this manner is like the expressions used by the `lme4` package in R where the triangularity of $\boldsymbol{L}$ is noted as simplifying calculations for linear systems (Doran, Bates, Bliese, & Dowling, 2007). Of course, this solution yields only one iteration and solutions for the mixed model iterate with updated values for the variance components. Note, that when all elements of the matrix $\boldsymbol{G} = \boldsymbol{0}$, this reduces to the partitioned fixed effects model.

Variance Estimation with Woodbury Identity and Decompositions for Matrix Inversions

Achieving scalability with iterative algorithms often requires finding less expensive ways to manage computations that are repeated. Linear mixed models are one such example where estimation depends on iterating between steps to compute variance components and then use those to estimate fixed parameters. It is one use case to further explore how better scalability can be achieved by

reducing the computational overhead and rewriting a problem to take advantage of an inversion lemma, a decomposition, and recycling certain components. In the linear mixed model, obtaining updated values for the variance components at each iteration of the algorithm can be estimated by first computing (McCulloch & Searle, 2001)

$$\boldsymbol{T} = \left[\boldsymbol{I} + \left(\boldsymbol{Z}'\boldsymbol{Z} - \boldsymbol{Z}'\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Z}\right)\boldsymbol{D}\right]^{-1} \tag{15}$$

where $\boldsymbol{I}$ is an identity matrix with dimensions equal to the model matrix for the random effects and $\boldsymbol{D} = \sigma_e^{-2}\boldsymbol{G}$ and all other matrices are as above. The variances at each iteration are computed (removing subscripts for iteration)

$$\sigma_e^2 = \frac{\boldsymbol{y}'\boldsymbol{e}}{N-p}, \quad \sigma_q^2 = \frac{\widetilde{\boldsymbol{u}}_q'\widetilde{\boldsymbol{u}}_q}{m_q - tr(\boldsymbol{T}_q)} \tag{16}$$

where σ_e^2 is the residual variance, σ_q^2 is the marginal variance at level q, $\widetilde{\boldsymbol{u}}_q$ are the predictions of the random effects at level q, m_q are the number of units in level q, and $tr(\boldsymbol{T}_q)$ is the trace of the matrix $\boldsymbol{T}_q$, which is the block of the matrix $\boldsymbol{T}$ corresponding to the qth block where a block is a submatrix within $\boldsymbol{T}$ generally formed from calculations involving $\boldsymbol{Z}_q'\boldsymbol{Z}_q$.

The challenge here is that the square matrix $\boldsymbol{T}$ can be very large. As a result, it is extremely expensive to find its inverse inside an iterative problem, thus rendering this approach computationally prohibitive unless a reduceable structure can be identified. In some cases, there is an alternative method for taking the inverse of a matrix with this structure via the Woodbury Matrix Identity (Woodbury, 1950) that provides a very efficient route for computation. First, Equation (15) is rewritten as

$$\boldsymbol{T} = \left(\boldsymbol{I} + \boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D} - \boldsymbol{Z}'\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Z}\boldsymbol{D}\right)^{-1} \tag{17}$$

where $\boldsymbol{D}$ is a diagonal matrix in the case of random intercepts and within each block q is constant along the diagonal. The Woodbury Identity provides that $\boldsymbol{T}$ can be computed as

$$\boldsymbol{T} = \boldsymbol{A}^{-1} + \boldsymbol{A}^{-1}\boldsymbol{B}\left(\boldsymbol{P} - \boldsymbol{B}'\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B}\right)^{-1}\boldsymbol{B}'\boldsymbol{D}\boldsymbol{A}^{-1} \tag{18}$$

where $\boldsymbol{A} = \boldsymbol{I} + \boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}$, $\boldsymbol{B} = \boldsymbol{Z}'\boldsymbol{X}$, and $\boldsymbol{P} = \boldsymbol{X}'\boldsymbol{X}$. Now using (18) $\boldsymbol{T}$ can be computed by separately finding the inverse of $\boldsymbol{A}$ and then also of $\boldsymbol{P} - \boldsymbol{B}'\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B}$. The latter of the two has very small dimensions, only $p \times p$ where p is the number of fixed effects in the model. The portion $\boldsymbol{I} + \boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}$ is a bit more cumbersome because the dimensions of the model matrix for the random effects can be extremely large. The Woodbury identity is useful when $\boldsymbol{A}^{-1}$ is available, thus simplifying the overall problem. We can now explore ways to efficiently find $\boldsymbol{A}^{-1}$ in this scenario.

The components $\boldsymbol{I}$ and $\boldsymbol{Z}'\boldsymbol{Z}$ are fixed and never change over the iterative process, only $\boldsymbol{D}$ is changing at each iteration. For nested random effects, $\boldsymbol{Z}'\boldsymbol{Z}$ is block diagonal but it has no special structure when the random effects are fully or partially crossed. This fact is motivation to explore if special structures can be exploited to more easily find $\boldsymbol{A}^{-1}$ given that it will be computed many times. Of course, it is a well-known result that the sum of inverses is not equal to the inverse of the sum, so finding $(\boldsymbol{I} + \boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D})^{-1}$ may be a hard problem. However, $\boldsymbol{I}$ and $\boldsymbol{D}$ are both diagonal within a block of q for nested random effects and have constant values along the diagonal. These structures make it possible to find $\boldsymbol{A}^{-1}$ in a trivial way for the linear mixed model when dealing with nested random effects.

One approach is to precompute and store the eigendecomposition for $\boldsymbol{Z}'\boldsymbol{Z}$, which is the most expensive part of the calculation, and rewrite the problem to diagonalize the matrix $\boldsymbol{A}$ to take advantage of simpler properties. These concepts give rise to (19), a full proof of which is provided in the appendix such that the inverse of the sum is then the inverse of the eigenvalues as

$$(\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D} + \boldsymbol{I})^{-1} = \boldsymbol{A}^{-1} = \boldsymbol{Q}_1\boldsymbol{\lambda}^{*-1}\boldsymbol{Q}_1' \tag{19}$$

which is trivial to obtain because $\boldsymbol{\lambda}^*$ is diagonal. Now $\boldsymbol{A}^{-1}$ is obtained in a much less expensive way by reusing $\boldsymbol{Q}_1$ and simply updating $\boldsymbol{\lambda}^*$ at each iteration. So, pulling together the Woodbury Identity and the decomposition derived for Equation (19), the inverse of the very big matrix $\boldsymbol{T}$ is easily computed repeatedly with little overhead.

This final example for linear models nicely illustrates all three concepts of precomputing, finding a reduceable structure, and using a numerically stable algorithm together during an iterative process to reduce computational overhead by many orders of magnitude. It's pointed out here that this specific rewrite works for special cases of the linear mixed model, such as nested random effects with random intercepts. However, the emphasis in this work is to identify special structures for a given scenario and exploit them to reduce computational burden. The inverse of the sum provided here may also have broader applications in other types of problems, such as when repeatedly finding the inverse of a Hessian matrix in iterative maximization problems, and so it is a motivating use case to document the condition.

Scalable Methods for Integration

Psychometricians routinely encounter challenging marginal likelihood problems or posterior distributions that require numerical integration methods. These methods approximate difficult integrals by summing over a finite number of points. Integrals with a gaussian factor are rather common in psychometric applications and the Gauss-Hermite rule (GHR) is then well-suited for integrals of the following form over the domain $(-\infty, \infty)$

$$\int f(\theta)e^{-\theta^2}d\theta \simeq \sum_{q=1}^{Q} f(\theta_q)w_q \tag{20}$$

where summation is over $\mathcal{P} = \{\theta_q, w_q\}_{q=1}^{Q}$, the so called nodes, $\theta_q \in \mathbb{R}^Q$, and weights, $w_q \in \mathbb{R}_+^Q$, for $q = 1, 2, \ldots, Q$. The choice of nodes and weights in $\mathcal{P}$ is optimal under GHR and will generally outperform other methods (Quarteroni, Saleri, & Gervasio, 2010). Unfortunately, psychometric practice seems to have codified the basic rectangular rule with fixed, equally spaced points from the domain of $(-4, 4)$ (Bock & Mislevy, 1982; DeMars, 2005) despite its serious limitations in most cases for at least two reasons. First, the equally spaced points may not be sampled in the best region of the integrand and second it lacks the computational advantage offered by advanced integration methods that rely on fewer, optimally chosen quadrature nodes. Building scalable applications requires approaches with minimal computational overhead that achieve precision rapidly. There are many other ways in which integrals can be evaluated. The motivation here is to focus on an efficient method that is generally useful for common psychometric scenarios that tends to scale well.

One-Dimensional Adaptive Gaussian Quadrature

Conveniently, psychometricians encounter scenarios where the weight function is a standard normal density

$$\int f(\theta)\frac{e^{-\theta^2/2}}{\sqrt{2\pi}}d\theta \simeq \pi^{-1/2}\sum_{q=1}^{Q} f(\sqrt{2}\boldsymbol{L}\theta_q + \widetilde{\theta})w_q \tag{21}$$

where $\widetilde{\theta}$ is a location parameter for centering (Liu & Pierce, 1994; Naylor & Smith, 1982) and $\boldsymbol{L}$ is the Cholesky factor of a covariance matrix. The $\boldsymbol{L}$ in (21) is actually a scalar, σ, but is left as a matrix so it appears as a special case of (23). The approximation in (21) implements a transformation on θ from (20) and adapts to the best region of the integrand—hence it is the adaptive GHR. Scalability and accuracy in evaluating integrals depends not only on the number of quadrature points, but also on the location of those points and the nature of the function surface being evaluated (Lesaffre & Spiessens, 2001).

The nodes and weights for the standard normal in the set $\mathcal{P}$ are generally available in standard tables (Abramowitz & Stegun, 1965) or commonly available in software packages, for example, the `statmod` package in R (Giner & Smyth, 2016). It is a useful connection to point out that the Golub-Welsch algorithm (Golub & Welsch, 1969) used for obtaining the nodes and weights in (21) is an eigendecomposition of a symmetric tridiagonal matrix providing the nodes as the resulting eigenvalues and weights as the squares of the first row of the eigenvectors. Sample R and Python code to obtain the nodes and weights using the Golub-Welsch algorithm is provided in the appendix.

High-Dimensional Adaptive Gaussian Quadrature

The one-dimensional GHR can be extended to the case for integrals of the general form in multiple dimensions

$$\int \dots \int f(\boldsymbol{\theta})\psi(\boldsymbol{\theta})d\boldsymbol{\theta}. \tag{22}$$

where $\boldsymbol{\theta} = \{\theta_1, \dots, \theta_p\}$ and continue with the assumption $\psi(\boldsymbol{\theta})$ is multivariate normal. The integral in (22) is often intractable and requires approximation which can be easily done with nodes transformed to reflect a covariance structure and adapted over a mean. That transformation can be implemented with the Cholesky factor of a covariance matrix as (Chowdhary, Salloum, Debusschere, & Larson, 2015; Jäckel, 2005; Judd, Maliar, & Maliar, 2011; Stringer, 2021)

$$\int \dots \int f(\boldsymbol{\theta})\psi(\boldsymbol{\theta})d\boldsymbol{\theta} \simeq \pi^{-P/2}\sum_{q_1=1}^{Q}\cdots\sum_{q_p=1}^{Q} f(\sqrt{2}\boldsymbol{L}(\theta_{q1}, \dots, \theta_{qp})' + (\widetilde{\theta}_{q1}, \dots, \widetilde{\theta}_{qp})')w_q^* \tag{23}$$

where θ_{qp} is node $q = 1, 2, \dots, Q$ in dimension $p = 1, 2, \dots P$, w_{qp} is the corresponding weight, $w_q^* = \prod_{p=1}^{P} w_{qp}$, and $\widetilde{\theta}_{qp}$ is used to mean some points around which the nodes are centered. The nodes are expansions of the points in the one-dimensional set, $\mathcal{P}$, via a cartesian product rule to form a multivariate grid, $\boldsymbol{G}$, with dimensions $Q^P \times P$. Hence, the qth row of $\boldsymbol{G}$ is $\boldsymbol{g}_q = (\theta_{q1}, \dots, \theta_{qp})$. Note that (21) is simply a special case of (23), hence no new concepts are needed for implementation beyond what has been described for the one-dimensional scenario. The

transformation in (21) and (23) yields nodes transformed from a unit normal to a normal with moments reflecting a mean $\widetilde{\boldsymbol{\theta}}$ and covariance $\boldsymbol{L}\boldsymbol{L}'$.

The extremely large number of quadrature points is what renders this approach challenging as, heuristically, this is effectively a two-step process. Step 1 requires evaluating the function $f(\cdot)$ over all nodes and then Step 2 is the summation of (23) making use of those values over all rows of the grid $\boldsymbol{G}$. Even with a modest selection for Q, the exponential growth in the function evaluations quickly becomes large with increasing P. Hence, exploring ways in which $f(\cdot)$ is evaluated as few times as possible and whether fewer function evaluations than $Q^P \times P$ can be used is the key to scalability as explored in the application section.

General Purpose Optimization Tools

Perhaps the most pervasive need for psychometric application building is a collection of tools for optimizing likelihoods. The likelihood theory of inference is well-developed (King, 1998) and found across all corners of psychometric practice. The common challenge is that likelihood functions for psychometric problems rarely have analytic forms and so iterative numerical approaches are used for maximum likelihood estimation (MLE). Two methods are explored here with a reminder that the stochastic gradient descent method is also a general purpose tool but was presented within the section on linear models and a growing number of optimization techniques becoming more widely used in machine learning can be further explored in a very friendly and readable resource provided by Rudner (2016).

Maximization with Newton-Raphson

Optimization problems in psychometrics generally involve a multi-parameter likelihood function intending to simultaneously maximize all parameters. In this instance, let $\mathcal{L}(\boldsymbol{\theta})$ denote a twice differentiable likelihood function with respect to the parameters $\boldsymbol{\theta}$ with gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$ and Hessian, $\mathcal{H}(\boldsymbol{\theta})$. The Newton-Raphson procedure maximizes $\mathcal{L}(\boldsymbol{\theta})$ by choosing initial starting values for $\boldsymbol{\theta}$ and then iteratively updating the parameters via

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \mathcal{H}_t(\boldsymbol{\theta})^{-1}\nabla\mathcal{L}_t(\boldsymbol{\theta}) \tag{24}$$

where the subscript t denotes the values of the gradient and Hessian at iteration t and continues until a convergence criterion is reached. The notation, $\mathcal{H}_t(\boldsymbol{\theta})^{-1}$, implies inversion of this matrix at each iteration. However, as previously discussed, the operation should be performed using a numerically stable decomposition. Additionally, because $\mathcal{H}_t(\boldsymbol{\theta})^{-1}$ is found iteratively and may be expensive, it may be feasible to find a trivial way to perform this operation like the eigendecomposition method in Equation (40) provided in the appendix that takes advantage of precomputing and reusing certain components. While the method in the appendix is unique to the mixed model scenario, it is intended to be a concept that can be generalized to other situations where a matrix inverse computed iteratively can be made less expensive. Quasi-Newton methods, such as the BFGS algorithm (Fletcher, 1987), are other less expensive options that are quite simple to implement and avoid directly computing $\mathcal{H}_t(\boldsymbol{\theta})^{-1}$, which may be difficult in some scenarios.

It can be pointed out that the SGD approach in (12) bears similarity to Newton's method of (24) where the learning rate of SGD is replaced by a Hessian in Newton's method. Conceptually,

Newton's method will more directly head towards the optimal value and SGD will essentially wiggle its way there. However, the cost of Newton's approach is much greater than that of SGD. Some work combining the faster convergence of Newton's method with the reduced computational complexity of SGD is available for further study (Sohl-Dickstein, Poole, & Ganguli, 2014; Zhou, Wei, Zhang, & Zheng, 2021).

Expectation-Maximization Algorithm

The expectation-maximization (EM) algorithm is broadly used across psychometrics in scenarios where the estimates desired are the MLEs of the observed data, but the observed data likelihood, $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x})$, is hard to maximize for various reasons (Dempster, Laird, & Rubin, 1977). If the gradient of the observed data likelihood is $\nabla\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x})$ then the optimization problem we would normally want consists of solving $\nabla\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}) = 0$. The EM algorithm simplifies this problem by imagining we can augment the observed data with other data, $\boldsymbol{\eta}$, to form complete data, $\boldsymbol{z} = \{\boldsymbol{x}, \boldsymbol{\eta}\}$, and a corresponding complete data likelihood, $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}) = \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{\eta})$, that can easily be maximized with respect to $\boldsymbol{\theta}$ if $\boldsymbol{z}$ were observed.

The problem is that only $\boldsymbol{x}$ are observed and $\boldsymbol{\eta}$ are latent and so maximizing $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z})$ cannot occur. Instead, $\boldsymbol{\eta}$ are simply assumed to be missing data and replaced with their conditional expected values to form the complete data. Then, the iterative approach outlined in Algorithm (3) can be used to alternate between an expectation step and a maximization step to find the MLEs. It can be pointed out that the Fisher identity (Fisher, 1925) provides that

$$\nabla\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}) = \int \nabla\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}) g(\boldsymbol{\eta}; \boldsymbol{x}, \boldsymbol{\theta}) d\boldsymbol{\eta} \tag{25}$$

which guarantees that the gradient of the observed data likelihood is the expectation of the gradient of the complete data likelihood integrated over the conditional distribution for the missing data (Cai, 2010). This implies that solving the right hand side of (25) is an alternative way to find the same parameter estimates that we would otherwise obtain if maximizing the observed data likelihood were possible.

Algorithm 3 Expectation Maximization Algorithm

Input: Create starting values for $\boldsymbol{\theta}$.

1: E-step: Compute expected values of $\boldsymbol{\eta}$ given the observed data and the provisional parameter estimate, $Q(\boldsymbol{\theta}; \boldsymbol{\theta}_t) = \mathbb{E}[\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{\eta})|\boldsymbol{x}, \boldsymbol{\theta}_t]$.
2: M-step: Find $\text{argmax}_{\boldsymbol{\theta}\in\boldsymbol{\Omega}}\, Q(\boldsymbol{\theta}; \boldsymbol{\theta}_t)$ given the provisional estimate, $\boldsymbol{\theta}_t$, where $\boldsymbol{\theta}$ is in the parameter space $\boldsymbol{\Omega}$.
3: Iterate between steps (1) and (2) until criterion for a stopping rule has been satisfied.

There are many examples of the EM algorithm in the psychometric literature for interested readers to further explore (Hsu, Ackerman, & Fan, 1999). One very simple example that can be quickly illustrated is to reconsider estimating the linear mixed model of the form $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{Z}\boldsymbol{u} + \boldsymbol{e}$. Here, the complete data are $\{\boldsymbol{y}, \boldsymbol{u}\}$ and if fully observed, the parameters of the mixed model could be found using Equation (16) for the variance components and Equation (8) for estimates of the fixed effects. The random effects are in reality missing, but the joint distribution of the

complete data, $\{\boldsymbol{y}, \boldsymbol{u}\}$, is assumed multivariate normal and standard distribution theory for the multivariate normal provides that $\mathbb{E}(\boldsymbol{u}|\boldsymbol{y})$ can be easily determined (McCulloch & Searle, 2001) and used to update a new value for the M-step.

It is somewhat challenging to use EM at scale without some additional considerations. For example, there are some occasions where the M-step may not have a closed form or data might be so large the maximization step could be daunting even when working values for the missing data are easily obtained. One very scalable option is to separate the computations when possible. For instance, there are occasions in psychometrics where the expected values used in the E-step can be computed independently, representing what is known as an embarrassingly parallel problem and the E-step (and possibly also the M-step) can be executed in parallel (Lee, Leemaqz, & McLachlan, 2016; Robitzsch, 2021; von Davier, 2016).

An interesting variant of the EM algorithm would be to use SGD in the M-step to manage a large data problem if the E-step was easily available. For example, in the case of the mixed model above, assume $\mathbb{E}(\boldsymbol{u}|\boldsymbol{y})$ is available through an efficient E-step, but maximization via (8) is hard because the dimensions of the model matrix $\boldsymbol{X}$ are extremely large. Instead of computing the M-step by (8), SGD could be used in the M-step to manage the process allowing computation to be highly scalable. An illustration for how this could be implemented for mixed models is provided in the appendix.

Applications to Psychometric Problems

The previously discussed methods are computational templates that can be used to build and scale psychometric applications. Three examples are provided here and the intent is not to center attention on these specific examples; rather, these examples are used because they illustrate how the preceding templates can be used to compute something complex that is not routinely found in standard software. For this reason, they are attractive for exploring computationally efficient approaches.

Example 1: Estimating the Error in Variables Linear Model

Estimates of examinee performance from a psychometric instrument are noisy reflections of a true, latent trait. The observed measures are subsequently used in various ways and often used as inputs in regression models of some form. When using these observed estimates, the assumptions of the Gauss-Markov theorem do not hold and the parameter estimates are then inconsistent unless corrections are made (Doran, 2014; Lockwood & McCaffrey, 2020; Nab, van Smeden, Keogh, & Groenwold, 2021).

These models are chosen as an example because with different assumptions about the nature of the measurement error (e.g., homoscedastic versus heteroscedastic), the models may take different forms and custom software is often written to obtain parameter estimates. One such example is the `mecor` (Nab et al., 2021) package in R and a code review of its implementation shows that the algebraic representation and computational representation are one and the same.

One form of the linear EiV regression is commonly shown to be (StataCorp, n.d.)

$$(\boldsymbol{X}'\boldsymbol{X} - \boldsymbol{S})\boldsymbol{\beta} = \boldsymbol{X}\boldsymbol{y} \tag{26}$$

where, in the case of known estimates for the reliability, the matrix $\boldsymbol{S} = diag(N(1-r_1)\sigma_1^2, \ldots, N(1-r_p)\sigma_p^2)$ where r_p is the reliability of the pth variable, N is the number of individuals, and σ_p^2 is the sample variance of the pth variable in the model matrix $\boldsymbol{X}$. Now, find $\boldsymbol{X}'\boldsymbol{X} - \boldsymbol{S} = \boldsymbol{L}\boldsymbol{L}'$ and obtain estimates using (8). It's useful to note that (26) can be extended for the linear mixed model and the computational approach previously shown can be applied.

Example 2: Examinee Ability Estimation Under the Correlated Factors Model

Suppose we are interested in estimating the ability of an examinee using a P-dimensional EAP estimator. This is chosen as an example because estimates from multidimensional IRT models such as the correlated factors model (Cai, 2010) are desirable estimators that continue to be challenging for psychometricians to implement (Chalmers, 2012; Ferrando & Lorenzo-Seva, 2016). Let the expected value of the posterior be

$$\widehat{\boldsymbol{\theta}} = \frac{\int \cdots \int \boldsymbol{\theta}\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z})\psi(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})d\boldsymbol{\theta}}{\int \cdots \int \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z})\psi(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})d\boldsymbol{\theta}} \tag{27}$$

where $\psi(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate normal density function and the likelihood function (see appendix) is a product of domain-specific likelihoods as $\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}) = \prod_{p=1}^{P} \mathcal{L}(\theta_p; \boldsymbol{z}_p)$. Individuals are assumed to have some idiosyncratic ability contributing to their response on each dimension and that some test items are uniquely associated with dimension 1 (θ_1), others with dimension 2 (θ_2), and so on for the Pth dimension, $\boldsymbol{\theta} = (\theta_1, \theta_2, \ldots, \theta_P)$.

The likelihood function and its corresponding population distribution in (27) are not in the same parametric family, as such the integral cannot be evaluated analytically. Here three options are explored that make for possible implementation that vary in the number of times the function $\mathcal{L}(\theta; \boldsymbol{z})$ is computed. Once the likelihood is evaluated, then step 2 is effectively direct implementation of the sum over all rows of $\boldsymbol{G}$, the matrix of quadrature nodes as previously described. Sparse grids or "pruning" some values are options for reducing the number of points over which the function is evaluated but those are not explored here.

The first canonical option is to use (23) directly. This involves using the Cholesky factor to adjust the nodes to absorb the population covariance structure and then, possibly shift the nodes over a new centered point. This works quite well; however, has one downside worth noting that makes this computationally somewhat less attractive.

Initially (before transforming with the Cholesky factor), the nodes in $\boldsymbol{G}$ are patterned such that marginally (i.e., each column in $\boldsymbol{G}$) the Q unique nodes are repeated over and over. When the rows of this matrix are transformed via $\boldsymbol{L}(\theta_{q1}, \ldots, \theta_{qp})'$, then the uniqueness is lost exponentially over each column. That is, column 1 has Q unique nodes repeated, column 2 has Q^2 unique nodes, column 3 would have Q^3 unique nodes and so on up to the Pth column.

This pattern of unique nodes creates a larger burden in that we must compute the values of the likelihood at each unique node and that IRT likelihood is somewhat expensive as it involves computing exponents. In column 1, it is feasible to compute the value of the likelihood at the Q unique points and then simply copy those values into the other positions in column 1 where the same value of the node appears. That is, we would compute the likelihood only Q times and then recycle those values by copying them into other positions where the same node appears in the grid. In column 2, we must compute the likelihood at all Q^2 unique nodes and then the values could then

be copied into the positions of this column where the same node appears again. The upside here is that the method yields a good approximation of the integral. The downside is the computational expense is very large. In this case, we would evaluate the likelihood $Q + Q^2 + Q^3, \ldots, Q^P$ times.

A second option for computing this integral could be to use a stochastic quadrature routine. In this process the nodes are random draws from $\mathcal{N}_p(\boldsymbol{M}, \boldsymbol{I}_P)$, where $\boldsymbol{I}_P$ is a P-dimensional identity matrix again creating the grid $\boldsymbol{G}$, but in this instance it would have dimensions $R \times P$ where R is the number of random variates chosen and it is not an exponential function of R. In this manner, the variates in $\boldsymbol{G}$ would reflect the population covariance after being premultiplied by the Cholesky factor (see introductory section) when the summation in (23) is used to evaluate the integral. This stochastic approach makes the computation reasonable and its precision is highly dependent on the law of large numbers such that $\widehat{\theta}_p \xrightarrow{a.s.} \theta_p$, $R \to \infty$, and also assuming the number of test items is very large. In this case, we would evaluate the likelihood $R \times P$ times.

The GHR tends to behave well when the function being evaluated approximates a low order polynomial (Liu & Pierce, 1994). However, in situations where an examinee responds to many test items, the function could spike in some areas causing for GHR to perform badly as few quadrature points could live within a region of interest. Perhaps the integral within (27) can be rewritten so that it behaves more like a low order polynomial (Antal & Oranje, 2007; Pinheiro & Bates, 1994; Rabe-Hesketh, Skrondal, & Pickles, 2002; Tuerlinckx, Rijmen, Verbeke, & De Boeck, 2006) and a third approach can be explored as

$$\int \cdots \int \mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}) \psi(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{\theta} = \int \cdots \int \frac{\mathcal{L}(\boldsymbol{\theta}; \boldsymbol{z}) \psi(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\widetilde{\psi}(\boldsymbol{\theta}; \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}})} \widetilde{\psi}(\boldsymbol{\theta}; \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}}) d\boldsymbol{\theta} \tag{28}$$

and then inserting back into (27) gives the approximation

$$\widehat{\theta}_p \simeq \frac{\sum_{q=1,1}^{Q} \cdots \sum_{q=p,1}^{Q} \theta_{p,q} \mathcal{L}(\theta_{1,q}; \boldsymbol{z}_1) \mathcal{L}(\theta_{2,q}; \boldsymbol{z}_2) \cdots \mathcal{L}(\theta_{P,q}; \boldsymbol{z}_p) \phi(\boldsymbol{\theta}_q; \boldsymbol{\mu}', \boldsymbol{\Sigma}') w_q^*}{\sum_{q=1,1}^{Q} \cdots \sum_{q=p,1}^{Q} \mathcal{L}(\theta_{1,q}; \boldsymbol{z}_1) \mathcal{L}(\theta_{2,q}; \boldsymbol{z}_2) \cdots \mathcal{L}(\theta_{P,q}; \boldsymbol{z}_p) \phi(\boldsymbol{\theta}_q; \boldsymbol{\mu}', \boldsymbol{\Sigma}') w_q^*} \tag{29}$$

where

$$\phi(\boldsymbol{\theta}_q; \boldsymbol{\mu}', \boldsymbol{\Sigma}') = \frac{\psi(\boldsymbol{\theta}_q; \boldsymbol{\mu}, \boldsymbol{\Sigma})}{\widetilde{\psi}(\boldsymbol{\theta}_q; \widetilde{\boldsymbol{\mu}}, \widetilde{\boldsymbol{\Sigma}})} \tag{30}$$

and $\widetilde{\psi}(\cdot)$ is a proposal distribution. This expression resembles an importance sampling approach (Gelman, Carlin, Stern, & Rubin, 2004) and has been termed importance Gauss-Hermite (IGH) method (Elvira et al., 2021). The value of the IGH approach is that the weights are retuned to have greater importance for certain nodes used to evaluate the integrand. The implication is that fewer quadrature points may be needed to evaluate the integral, thus leading to greater scalability in high-dimensional psychometric problems. The trade-off is that some calculations involving the densities in $\phi(\cdot)$ is required, though that additional cost is seemingly smaller than if more quadrature points were used and importance-based methods may be valuable for reducing computational burden (Ackerberg, 2009).

The additional benefit is that the original (unrotated) Gauss-Hermite nodes are used in (29). We can exploit the uniqueness in the repeating values of the nodes in each column and compute the value of the likelihood only Q times in each dimension and then copy those values into other positions of the column where they repeat. As a result, the likelihood is evaluated only $Q \times P$

times and not $Q^P \times P$ times! Assuming $Q < R$, this approach evaluates the likelihood the smallest number of times of all three cases.

Some additional computational benefits could easily be realized for this problem. For instance, computing the likelihood values over each column in the node array is independent and consequently is embarrassingly parallel, and the results could then be joined to take summations. In fact, test scoring applications in psychometrics are an embarrassingly parallel problem as the test score for an examinee does not depend on the scores for others, it depends only on the item parameters and their item responses, both of which are treated as fixed.

Generally, simple rectangular rules are less precise than the methods described here but there may be occasions where they offer some computational advantages. One possible benefit of using a fixed quadrature grid is that calculations involving the likelihood can be performed once, stored, and recycled. In scenarios where nodes are changing in an iterative process, this would require computing the likelihood over and over with the evolving nodes and that computation is expensive. Examples making use of this concept may be found in the `Dire` package in `R` (Doran et al., 2021) and also in Robitzsch (2021).

Example 3: Supervised Learning with SGD

Imagine a group of test item developers would benefit from knowing whether the new items being developed are likely to have examinee response times that are fast or slow before those items have any field test and timing statistics. Such forecasting systems could guide item development plans and item writing workshops and be helpful in an array of test construction activities.

Assume response time data from a testing program exist for each test item and the data could be concatenated over years, over grades, over test forms, and over all examinees yielding an extremely large data file forming n total observations. Further assume all items in the set have a collection of p observable features that are predictive of response time, such as readability indices, stem word length, content domain, cognitive complexity, item type and so on. The outcome data are binary outcomes associated with item response time such that $y_i = 1$ if the response to the ith item is fast and $y_i = 0$ otherwise and that the observable features are in the model matrix $\boldsymbol{X}$. Response time is of course a real positive number, but here we bin into two categories to build an illustration that differs from the prior linear regression model using SGD.

The example described here yields an extremely large data file from which we intend to learn patterns and then make future predictions on other data not included in the training sample. Because the available data to be used for training is so large, logistic regression using something like iteratively reweighted least squares is not feasible. In this case, a simple supervised learning problem can be constructed using SGD for training allowing for computation to occur in an efficient way.

Normally, batch gradient descent with n observations, p parameters, and using k epochs would evaluate the gradients $n \times k \times p$ times. The number for k is generally arbitrary but is often selected as a very large value. Instead, SGD can reduce this computational burden and evaluate only $k \times p$ gradient terms to achieve comparable results. In fact, because SGD randomly samples from the full set of observations, it may be helpful to randomly stream in portions of the full available data and sample from within those portions repeatedly. This could alleviate some overhead space needed to be reserved for computing. If mini-batch gradient descent were used, then $n_i \times k \times p$ gradient terms would be evaluated, where n_i represents a subset of n. The reduction in the

number of gradient evaluations is impressive and makes very large problems feasible. Simply to be conceptual, imagine $n = 1,000,000$, $k = 1000$, and $p = 3$; then, batch gradient descent would evaluate 3 billion gradient terms whereas SGD would only evaluate 3,000 gradient terms!

Here, let the objective function for the logistic model be $\mathcal{J}_i(\boldsymbol{\theta}) = -y_i \log(z_i) + (1 - y_i)(1 - z_i)$ and then

$$\nabla \mathcal{J}_i(\boldsymbol{\theta}) = (z_i - y_i)\boldsymbol{X}_i \tag{31}$$

where the following familiar logistic (sigmoid) function is used

$$\Pr(y_i = 1|, \boldsymbol{X_i}, \boldsymbol{\beta}) = z_i = \frac{1}{1 + \exp(-(\boldsymbol{X}_i\boldsymbol{\beta}))}. \tag{32}$$

The gradient in (31) can be used in Equation (12) and the steps in Algorithm (1) would be implemented for optimization. Generalizations of this concept could be further developed to explore IRT forecasting problems where a sense of the future item parameters could be obtained from features associated with items or in training a learning management system to associate targeted instructional content to an examinee based on their test performance.

Discussion

This paper has presented ways in which psychometricians may build efficient and scalable applications with large, complex data. Four general topics are explored including whitening transformations useful for correlated data and computational concepts needed for numerically stable and scalable linear models, multivariable integration, and optimization methods. The ideas here are but a subset of useful numerical methods. However, the subset represents a core set of methods that can be combined or generalized and broadly applied in psychometric application building. Hopefully, this collection offers some advancement in supporting scalable computational methods necessary to support the future challenges of computationally demanding psychometric models.

There are a few possible ways in which the work described here might be useful. First, many psychometricians find jobs in industry where they are charged with creating psychometric applications to support their organizational infrastructure and client requirements. The applications used by assessment companies deploy tests to many hundreds of thousands (even millions) of examinees and the operational systems used for psychometric work must be fast, accurate, and highly scalable. Second, perhaps in academia, the ideas here can supplement other course materials borrowed from computational statistics that are more mathematical in nature. This work is intended to create connections to computational psychometrics and may be a helpful way to bridge the knowledge base between general numerical methods and psychometric theory. Third, many commercial psychometric applications need some retrofitting to better support larger computational demands and the ideas here can be potentially used to support those upgrades. Last, psychometricians setting out on their own course of study to improve their computational acumen might benefit from implementing the ideas proposed here as test cases to explore and replicate.

The references offered in this manuscript are valuable extensions to the topics explored here and readers interested in advancing their acumen in this area are encouraged to review them in detail, many of which are drawn from disciplines outside the field of psychometrics. Additionally, some extremely helpful massive open online courses (MOOCs) exist on topics in numerical analysis and machine learning offered by the Massachusetts Institute of Technology, Harvard EDX,

and Coursera (e.g., Numerical Methods for Engineers). While the field is expanding very rapidly, many helpful resources exist to support continued learning. However, as stated from the onset of this paper, the advancements are generally expanding around many of the first principles described within this paper. Hence, fully appreciating the subset of methods discussed here is a significant way to stay positioned for the oncoming methodological advances of the future.

Finally, it is interesting to imagine the future of psychometrics and the general skill set that may be needed for future psychometricians. While psychometricians must continue to be domain experts in measurement, the work of measurement is now more explicitly integrated with the development of computational infrastructure. This implies some expertise not only in the topics discussed in this paper, but also in the types of skills commonly learned in computer science programs. For instance, being able to build application programming interface (API) endpoints, or at least engage meaningfully in the design of the API as a team member, understanding cloud-based infrastructures and how they scale (e.g., AWS, Google Cloud, Microsoft Azure), some skills related to full stack software development, containers such as Docker, understanding the interoperability data models espoused by IMS Global (e.g., QTI, XML, JSON) are virtually daily conversations for psychometricians in industry and topics for future psychometricians to be very familiar with.

Appendix A

Proof that $(\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}+\boldsymbol{I})^{-1}=\boldsymbol{Q}_1\boldsymbol{\lambda}^{*-1}\boldsymbol{Q}_1'$

Let the eigendecompositions be $\boldsymbol{Z}'\boldsymbol{Z}=\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{Q}_1'$ and $\boldsymbol{D}=\boldsymbol{Q}_2\boldsymbol{\lambda}_2\boldsymbol{Q}_2'=\boldsymbol{\lambda}_2$ given that $\boldsymbol{D}=d\boldsymbol{I}$ where the $\boldsymbol{Q}$s contain the eigenvectors and the $\boldsymbol{\lambda}$s contain the eigenvalues, then

$$\begin{aligned}\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}+\boldsymbol{I}&=(\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{Q}_1')(\boldsymbol{Q}_2\boldsymbol{\lambda}_2\boldsymbol{Q}_2')+\boldsymbol{I} &(33)\\&=(\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{Q}_1')\boldsymbol{\lambda}_2+\boldsymbol{I} &(34)\\&=\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2\boldsymbol{Q}_1'+\boldsymbol{I} &(35)\end{aligned}$$

given that the commutative property for matrices holds iff $d\boldsymbol{I}$, then $\boldsymbol{Q}_1'\boldsymbol{\lambda}_2\iff\boldsymbol{\lambda}_2\boldsymbol{Q}_1'$. Then

$$\begin{aligned}\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2\boldsymbol{Q}_1'+\boldsymbol{I}&=\boldsymbol{Q}_1(\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2+\boldsymbol{I})\boldsymbol{Q}_1' &(36)\\&=\boldsymbol{Q}_1\boldsymbol{\lambda}^*\boldsymbol{Q}_1' &(37)\end{aligned}$$

where $\boldsymbol{\lambda}^*=diag\{\lambda_{11}d+1,\lambda_{12}d+1,\ldots,\lambda_{1n}d+1\}$ and λ_{1n} are the elements of $\boldsymbol{\lambda}_1$. Moving $\boldsymbol{I}$ in (36) is easily verified by expanding

$$\begin{aligned}\boldsymbol{Q}_1(\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2+\boldsymbol{I})\boldsymbol{Q}_1'&=\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2\boldsymbol{Q}_1'+\boldsymbol{Q}_1\boldsymbol{I}\boldsymbol{Q}_1' &(38)\\&=\boldsymbol{Q}_1\boldsymbol{\lambda}_1\boldsymbol{\lambda}_2\boldsymbol{Q}_1'+\boldsymbol{I} &(39)\end{aligned}$$

in which case $\boldsymbol{Q}_1\boldsymbol{I}\boldsymbol{Q}_1'=\boldsymbol{I}$ when $\boldsymbol{Q}_1'$ is orthonormal. Finally, the inverse of the sums is simplified to the inverse of the eigenvalues

$$(\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}+\boldsymbol{I})^{-1}=\boldsymbol{Q}_1\boldsymbol{\lambda}^{*-1}\boldsymbol{Q}_1'. \quad (40)$$

Equation (40) is useful in iterative algorithms as it precomputes an expensive component, $\boldsymbol{Q}_1$, initially and then it is repeatedly reused in subsequent iterations. Because $\boldsymbol{\lambda}^*$ is diagonal, it permits a trivial inverse.

General Remark on the Proof

It is important to note that the preceding proof applies under conditions when $\boldsymbol{D}=d\boldsymbol{I}$, in which case the commutative property holds between $\boldsymbol{D}$ and some other (conformable) matrix. However, when that condition is not true, we arrive at a slightly different problem. Suppose $\boldsymbol{D}$ is diagonal, but the elements along the diagonal are not constant. Then, we may write $(\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}+\boldsymbol{I})^{-1}=(\boldsymbol{Q}\boldsymbol{\lambda}\boldsymbol{Q}'+\boldsymbol{I})^{-1}=\boldsymbol{Q}(\boldsymbol{\lambda}+\boldsymbol{I})^{-1}\boldsymbol{Q}'$ where now $\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}=\boldsymbol{Q}\boldsymbol{\lambda}\boldsymbol{Q}'$.

This is interesting, but not entirely helpful within an iterative algorithm as $\boldsymbol{D}$ cannot be factored out of the decomposition. Hence, it requires computing the decomposition $\boldsymbol{Z}'\boldsymbol{Z}\boldsymbol{D}$ at each iteration even though only $\boldsymbol{D}$ is changing. At the current time, there is no accepted method for the inverse of matrix sums under this condition. There are some concepts that may later be considered using eigenvalue perturbations. That is, if we can first compute the decomposition $\boldsymbol{Z}'\boldsymbol{Z}$, then update this with the changing values of $\boldsymbol{D}$ to yield an approximation of $\boldsymbol{Q}\boldsymbol{\lambda}\boldsymbol{Q}'$, then we might be able to efficiently find the sum of the inverse when $\boldsymbol{D}$ is diagonal, but not with constant elements. This remains a numerical challenge to explore.

Likelihood Function Details

Each individual domain likelihood, $\mathcal{L}(\theta_P; \boldsymbol{z}_p)$, is composed of a mixture of binary and polytomous test items with known item parameters. Let z_{ijp} denote the observed response of the ith examinee to the jth item in the pth dimension, then

$$\mathcal{L}(\theta_P; \boldsymbol{z}_p) = \mathcal{L}_1(\theta_P; \boldsymbol{z}_p)\mathcal{L}_2(\theta_P; \boldsymbol{z}_p)$$

$$\mathcal{L}_1(\theta_P; \boldsymbol{z}_p) = \prod_{j \in p} \left[c_j + \frac{1 - c_j}{1 + \exp[-Da_j(\theta_P - b_j)]} \right]^{z_{ijp}} \left[1 - \left(c_j + \frac{1 - c_j}{1 + \exp[-Da_j(\theta_P - b_j)]} \right) \right]^{1 - z_{ijp}}$$

where $j \in p$ is used to mean there is a collection of j items uniquely associated with dimension p, c_j is the lower asymptote of the item response curve (i.e., the guessing parameter), a_j is the slope of the item response curve (i.e., the discrimination parameter), $b_j$is the location parameter, and D is a constant, by default fixed at 1.7. Then the items scored in multiple categories takes the form of the graded response model as

$$\mathcal{L}_2(\theta_P; \boldsymbol{z}_p) = \prod_{j \in p} \Pr(z_{ijp}|\theta_P)$$

$$\Pr(z_{ijp}|\theta) = \begin{cases} \frac{1}{1+e^{Da_j(\theta_P - b_{j1})}}, z_{ijp} = 0 \\ \frac{1}{1+e^{Da_j(\theta_P - b_{j,z+1})}} - \frac{1}{1+e^{Da_j(\theta_P - b_{jz})}}, \ 0 < z_{ijp} < K \\ \frac{1}{1+e^{-Da_j(\theta_P - b_{jK})}}, \ z_{ijp} = K \end{cases} \tag{41}$$

where b_K denotes the kth step and the other definitions used for the binary model apply here.

Efficient Parallel E-Step Example for Linear Mixed Model

Let the joint distribution of the fixed and random effects be written as a multivariate normal

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{y} \end{bmatrix} \sim \mathcal{N} \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{X\beta} \end{bmatrix} \begin{bmatrix} \boldsymbol{G} & \boldsymbol{GZ'} \\ \boldsymbol{ZG} & \boldsymbol{V} \end{bmatrix} \tag{42}$$

where $\boldsymbol{V} = \boldsymbol{ZGZ'} + \boldsymbol{\Omega}$. Standard distribution theory provides that the conditional values of the random effects from this joint multivariate normal are $\mathbb{E}[\boldsymbol{u}|\boldsymbol{y}] = \boldsymbol{GZ'V}^{-1}(\boldsymbol{y} - \boldsymbol{X\beta})$. For two-level nested models with random intercepts only, this can be written as follows for the jth group with N_j denoting the number of units in the group

$$\mathbb{E}[\boldsymbol{u}_j|\boldsymbol{y}_j] = \widetilde{\boldsymbol{u}}_j = \left[\frac{\sigma_q^2}{\sigma_e^2 + N_j\sigma_q^2} \right] \sum_{i \in j} (y_i - \boldsymbol{X}_i\boldsymbol{\beta}) \tag{43}$$

where $\boldsymbol{X}_i$ is the ith row in the model matrix $\boldsymbol{X}$ and σ_e^2 and σ_q^2 are the residual variance and the marginal variance of the random effects at level q, respectively. Equation (43) can be computed as an embarrassingly parallel problem for each of the j groups. Assembling all $\widetilde{\boldsymbol{u}}_j$ to use as the provisional working values for the missing data, we can now proceed with the M-step using stochastic gradient descent for least squares as shown in the section on linear models.

Appendix B

```r
gaussHermiteNorm <- function(Q){
  y <- sqrt(1:(Q-1))
  m <- diag(0, Q)
  m[row(m) - col(m) == 1] <- m[row(m) - col(m) == -1] <- y
  result <- eigen(m)
  list(nodes = result$values, weights = result$vector[1,]^2)
}
```

```python
def gauss_quad_normal(Q, mu = 0, sigma = 1):
    y = np.sqrt(range(1,Q))
    m = np.zeros((Q,Q))
    ind = np.arange(Q-1)
    m[ind,ind+1] = y
    m[ind+1,ind] = y
    result = np.linalg.eig(m)
    nodes = result[0] * sigma + mu
    weights = result[1][0]**2
    return nodes, weights

def gauss_grid(Q, mu = 0, sigma = 1, dimensions = 1):
    nodes, weights = gauss_quad_normal(Q)
    result = np.meshgrid(*[nodes] * dimensions,sparse=False,
       indexing='ij')
    nodesArray = [[None for y in range(Q**dimensions)] for x in
       range(dimensions)]
    for i in range(dimensions):
        nodesArray[i] = result[i].flatten('F')
    weightList = np.meshgrid(*[weights] * dimensions, sparse=False,
       indexing='ij')
    weightsArray = [[None for y in range(Q**dimensions)] for x in
       range(dimensions)]
    for i in range(dimensions):
        weightsArray[i] = weightList[i].flatten('F')
    final_weights = [0.] * Q**dimensions
    for i in range(dimensions):
        final_weights +=  np.log(weightsArray[i])
    final_weights = np.exp(final_weights)
    return gridArray, final_weights
```

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., ... Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `https://www.tensorflow.org/` (Software available from tensorflow.org)

Abramowitz, M., & Stegun, I. A. (Eds.). (1965). *Handbook of mathematical functions with formulas, graphs and mathematical tables*. New York: Dover Publications, Inc.

Ackerberg, D. (2009). A new use of importance sampling to reduce computational burden in simulation estimation. *Quant Mark Econ*, *7*, 343-376.

Allen, N., Donoghue, J., & Schoeps, T. (2001). *The NAEP 1998 technical report* (Tech. Rep.). U.S. Department of Education. Office of Educational Research and Improvement. National Center for Education Statistics. Retrieved from `https://nces.ed.gov/nationsreportcard/pubs/main1998/2001509.asp`

Andersson, B., & Xin, T. (2021). Estimation of latent regression item response theory models using a second-order laplace approximation. *Journal of Educational and Behavioral Statistics*, *46*(2), 244-265.

Antal, T., & Oranje, A. (2007). *Adaptive numerical integration for item response theory* (Tech. Rep.). Educational Testing Service. Retrieved from `https://files.eric.ed.gov/fulltext/EJ1111562.pdf`

Bates, D. (2004a). Least squares calculations in R. *R News*, *4*(1), 17–20.

Bates, D. (2004b). *Sparse matrix representations of linear mixed models.* Retrieved from `http://www.stat.wisc.edu/~bates/reports/MixedEffects.pdf`

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. Retrieved from `https://epubs.siam.org/doi/10.1137/141000671` doi: 10.1137/141000671

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement*, *6*(4), 431-444.

Cai, L. (2010). Metropolis-hastings robbins-monro algorithm for confirmatory item factor analysis. *Journal of Educational and Behavioral Statistics*, *35*(3), 307-335.

Cai, L., Yang, J., & Hansen, M. (2011). Generalized full-information item bifactor analysis. *Psychological methods*, *16*(3), 221-248.

Chalmers, R. (2012). Mirt: A multidimensional item response theory package for the R environment. *JSS Journal of Statistical Software*, *48*.

Chowdhary, K., Salloum, M., Debusschere, B., & Larson, V. E. (2015). Quadrature methods for the calculation of subgrid microphysics moments. *Monthly Weather Review*, *143*(7), 2955-2972.

Cizek, P., & Cizkova, L. (2004). Iterative methods for solving linear systems. In J. Gentle, W. Härdle, & Y. Mori (Eds.), *Handbook of computational statistics* (p. 120-126). New York: Springer.

Davis, T., & Hager, W. (2005). Row modifications of a sparse cholesky factorization. *SIAM J. Matrix Analysis Applications*, *26*, 621-639.

DeMars, C. E. (2005). Scoring subscales using multidimensional item response theory models. Washington, DC: Poster presented at the annual meeting of the American Psychological Association.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, *39*, 1-38.

Doran, H. (2014). Methods for incorporating measurement error in value-added models and teacher classifications. *Statistics and Public Policy*, *1*(1), 114-119.

Doran, H., Bailey, P., Buehler, E., & joo Lee, S. (2021). Dire: Linear regressions with a latent outcome variable [Computer software manual]. Retrieved from `https://CRAN.R-project.org/package=Dire` (R package version 1.0.3)

Doran, H., Bates, D., Bliese, P., & Dowling, M. (2007). Estimating the multilevel rasch model: With the lme4 package. *Journal of Statistical Software*, *20*(2), 1-18. Retrieved from `https://www.jstatsoft.org/index.php/jss/article/view/v020i02` doi: 10.18637/jss.v020.i02

Elvira, V., Martino, L., & Closas, P. (2021). Importance gaussian quadrature. *IEEE Transactions on Signal Processing*, *69*, 474-488. Retrieved from `http://dx.doi.org/10.1109/TSP.2020.3045526`

Ferrando, P., & Lorenzo-Seva, U. (2016). A note on improving EAP trait estimation in oblique factor-analytic and item response theory models. *Psicologica*, *37*(2), 235-247.

Fisher, R. A. (1925). Theory of statistical estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, *22*(5), 700-725. doi: 10.1017/S0305004100009580

Fletcher, R. (1987). *Practical methods of optimization* (Second ed.). New York, NY, USA: John Wiley & Sons.

Gelman, A., Carlin, J. B., Stern, H. S., & Rubin, D. B. (2004). *Bayesian data analysis* (2nd ed. ed.). Chapman and Hall/CRC.

Genz, A., & Kass, R. E. (1997). Subregion-adaptive integration of functions having a dominant peak. *Journal of Computational and Graphical Statistics*, *6*(1), 92-111.

Giner, G., & Smyth, G. K. (2016). statmod: Probability Calculations for the Inverse Gaussian Distribution. *The R Journal*, *8*(1), 339–351. Retrieved from https://doi.org/10.32614/RJ-2016-024 doi: 10.32614/RJ-2016-024

Golub, G., & Welsch, J. (1969). Calculation of gauss quadrature rules. *Mathematics of Computation*, *23*, 221-230.

Hedges, L. V., & Hedberg, E. C. (2007). Intraclass correlation values for planning group-randomized trials in education. *Educational Evaluation and Policy Analysis*, *29*(1), 60-87. Retrieved from https://doi.org/10.3102/0162373707299706 doi: 10.3102/0162373707299706

Hsu, Y., Ackerman, T. A., & Fan, M. (1999). The relationship between the Bock-Aitkin procedure and the EM algorithm for IRT model estimation.. Retrieved from https://www.act.org/content/dam/act/unsecured/documents/ACT_RR99-07.pdf

Innes, M., Edelman, A., Fischer, K., Rackauckas, C., Saba, E., Shah, V. B., & Tebbutt, W. (2019). A differentiable programming system to bridge machine learning and scientific computing. *CoRR*, *abs/1907.07587*. Retrieved from http://arxiv.org/abs/1907.07587

Jäckel, P. (2005). *A note on multivariate gauss-hermite quadrature.* Retrieved from http://www.jaeckel.org/ANoteOnMultivariateGaussHermiteQuadrature.pdf

Judd, K. L., Maliar, L., & Maliar, S. (2011). Numerically stable and accurate stochastic simulation approaches for solving dynamic economic models. *Quantitative Economics*, *2*(2), 173-210.

Kessy, A., Lewin, A., & Strimmer, K. (2018). Optimal whitening and decorrelation. *The American Statistician*, *72*(4), 309-314.

King, G. (1998). *Unifying political methodology: The likelihood theory of statistical inference*. Ann Arbor: University of Michigan Press.

Koseoglu, B. (2018). *Understanding ordinary least square in matrix form with R.* Retrieved from https://medium.com/@bengikoseoglu/understanding-ordinary-least-square-in-matrix-form-with-r-b6cf2d08a93b

Laird, N. M., & Ware, J. H. (1982). Random-effects models for longitudinal data. *Biometrics*, *38*(4), 963–974.

Lee, S. X., Leemaqz, K. L., & McLachlan, G. J. (2016). A simple parallel EM algorithm for statistical learning via mixture models. In *2016 international conference on digital image computing: Techniques and applications (dicta)* (p. 1-8).

Lesaffre, E., & Spiessens, B. (2001). On the effect of the number of quadrature points in a logistic random-effects model: An example. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *50*(3), 325–335.

Lira, M., Iyer, R., Trindade, A., & Howle, V. (2016). QR versus cholesky: A probabilistic analysis. *International Journal of Numerical Analysis and Modeling*, *13*(1), 114–121. (Publisher Copyright: © 2016 Institute for Scientific Computing and Information.)

Liu, Q., & Pierce, D. A. (1994). A note on gauss-hermite quadrature. *Biometrika*, *81*(3), 624–629.

Lockwood, J. R., & McCaffrey, D. F. (2020). Recommendations about estimating errors-in-variables regression in stata. *The Stata Journal*, *20*(1), 116-130.

McCulloch, C. E., & Searle, S. R. (2001). *Generalized, linear, and mixed models*. New York: John Wiley and Sons.

McLean, R. A., Sanders, W. L., & Stroup, W. W. (1991). A unified approach to mixed linear models. *The American Statistician*, *45*(1), pp. 54-64.

Mislevy, R. J. (1984). Estimating latent distributions. *Psychometrika*, *49*(3), 359-381.

Muraki, E., & Bock, R. D. (1999). Parscale: IRT item analysis and test scoring for rating-scale data [Computer software manual]. Chicago, IL.

Nab, L., van Smeden, M., Keogh, R. H., & Groenwold, R. H. (2021). Mecor: An R package for measurement error correction in linear regression models with a continuous outcome. *Computer Methods and Programs in Biomedicine*, *208*, 106238.

National Research Council. (2013). *Frontiers in massive data analysis*. Washington, DC: The National Academies Press. ("https://www.nap.edu/catalog/18374/frontiers-in-massive-data-analysis) doi: 10.17226/18374

Naylor, J. C., & Smith, A. M. (1982). Applications of a method for the efficient computation of posterior distributions. *Journal of The Royal Statistical Society Series C-applied Statistics*, *31*, 214-225.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

Pinheiro, J., & Bates, D. (1994). Approximations to the log-likelihood function in the nonlinear mixed-effects model. *Journal of Computational and Graphical Statistics*, *4*. doi: 10.1080/10618600.1995.10474663

Quarteroni, A., Saleri, F., & Gervasio, P. (2010). *Scientific computing with matlab and octave. 3rd ed*. Springer.

Rabe-Hesketh, S., Skrondal, A., & Pickles, A. (2002). Reliable estimation of generalized linear mixed models using adaptive quadrature. *The Stata Journal*, *2*(1), 1-21.

Rijmen, F. (2009). *Efficient full information maximum likelihood estimation for multidimensional irt models* (Tech. Rep.). Educational Testing Service. Retrieved from https://files.eric.ed.gov/fulltext/ED505564.pdf (Research Report No. RR-09-03)

Robitzsch, A. (2021). A note on a computationally efficient implementation of the EM algorithm in item response models. *Quantitative and Computational Methods in Behavioral Sciences*, *1*(1). doi: 10.5964/qcmb.3783

Rosseel, Y. (2021). Evaluating the observed log-likelihood function in two-level structural equation modeling with missing data: From formulas to R code. *Psych*, *3*(2), 197–232.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv*, *abs/1609.04747*.

SAS Documentation 14.2. (n.d.). Estimating fixed and random effects in the mixed model [Computer software manual]. (Online Help Manual)

Searle, S. (1982). *Matrix algebra useful for statistics*. New York: John Wiley and Sons.

Shamir, O., & Zhang, T. (2013). Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning* (Vol. 28, pp. 71–79). Atlanta, Georgia, USA: PMLR. Retrieved from https://proceedings.mlr.press/v28/shamir13.html

Shustin, P. F., & Avron, H. (2021). *Semi-infinite linear regression and its applications.* arXiv. Retrieved from https://arxiv.org/abs/2104.05687 doi: 10.48550/ARXIV.2104.05687

Sohl-Dickstein, J., Poole, B., & Ganguli, S. (2014). Fast large-scale optimization by unifying stochastic gradient and quasi-newton methods. In *Proceedings of the 31th international conference on machine learning* (Vol. 32). Beijing, China. Retrieved from http://proceedings.mlr.press/v32/sohl-dicksteinb14.pdf

StataCorp. (n.d.). Errors-in-variables regression [Computer software manual]. Retrieved from https://www.stata.com/manuals/reivreg.pdf (Online Help Manual)

Stringer, A. (2021). *Implementing approximate bayesian inference using adaptive quadrature: the aghq package.*

Teng, S.-H. (2016). Scalable algorithms for data and network analysis. *Foundations and Trends in Theoretical Computer Science*, *12*(1-2), 1-274. Retrieved from http://dx.doi.org/10.1561/0400000051 doi: 10.1561/0400000051

Tran, D., Toulis, P., & Airoldi, E. (2015). Stochastic gradient descent methods for estimation with large data sets. *Journal of Statistical Software*.

Tuerlinckx, F., Rijmen, F., Verbeke, G., & De Boeck, P. (2006). Statistical inference in generalized linear mixed models: A review. *The British journal of mathematical and statistical psychology*, *59*, 225-55.

von Davier, M. (2016). High-performance psychometrics: The parallel-e parallel-m algorithm for generalized latent variable models. *ETS Research Report Series*, *2016*, 1-11.

Woodbury, M. A. (1950). Inverting Modified Matrices. In J. Kuntzmann (Ed.), . Princeton, NJ: Princeton University.

Zhang, F. (1999). *Matrix theory: Basic results and techniques*. New York: Springer.

Zhou, J., Wei, W., Zhang, R., & Zheng, Z. (2021). Damped newton stochastic gradient descent method for neural networks training. *Mathematics*, *9*(13). Retrieved from https://www.mdpi.com/2227-7390/9/13/1533