Measurement and Research Department Reports 2005-4

Algorithms for Parameter Estimation in the Rasch Model

Joost van Ruitenburg

CITO

Measurement and Research Department Reports 2005-4

Algorithms for Parameter Estimation in the Rasch Model

Master's Thesis in Econometrics

Joost van Ruitenburg

Erasmus University Rotterdam, and
Psychometric Research Center
CITO, Dutch National Institute of Educational Measurement

Supervisors

Prof. dr. P.J.F. Groenen	Erasmus University Rotterdam
Dr. J. Brinkhuis	Erasmus University Rotterdam
Dr. T.M. Bechger	Cito
Dr. G.K.J. Maris	Cito

Cito
Arnhem, 2005

Preface

It all started on the 16th of September 2004. I read an advertisement in the Erasmus Magazine about a internship at Cito, which seemed interesting. This company does a lot of research in the field of Psychometrics, and I knew that prof. Groenen is also acquainted with this field. He was positive about the research of the company, and so I contacted prof. Sanders at Cito. Before I realized it, I had signed a contract to write my thesis there, starting in April.

This proved to be a good decision. I was given my own room, and was supervised by two well-learned enthusiastic doctors, Timo Bechger and Gunter Maris. At a suggestion made by prof. Groenen, we decided to use techniques from his area of research for the parameter estimation of models, which are interesting for CITO.

In the first months, I was busy programming all kinds of existing and self-invented algorithms in Visual Basic. In the meanwhile, we were having theoretical discussions on alternative methods, and their characteristics. After a couple of months, we had to make the decision whether to expand the research or to go deeper into the the methods we were already working with. Looking back, I think we made the right decision to choose the latter option.

Meanwhile, prof. Groenen had given his consent to write a thesis on this topic. In the summer I did some simulations and wrote this piece. Two fellow students, Ferdinand Houwink and Anna Lim, were kind enough to read the thesis, and to point out some ambiguities. Finally, my supervisors gave some final remarks in September and in October, after which I completed this thesis. I want to thank all persons whom I have previously named for their help. I hope the reader will become (more) interested in the world of algorithms after reading this thesis.

Abstract

We compare several methods to find the parameters of the Rasch Model, well-known in psychometrics. This search entails finding the maximum of the concave conditional log-likelihood function. We focus on iterative minorization and interval algorithms, which update one parameter per iteration, and are guaranteed to converge. We present six algorithms, an acceleration method, and perform a small simulation study with them. This study shows that the convergence time of the algorithms depends on the shape of the likelihood function.

Contents

1. Introduction

As the name suggests, *Maximum Likelihood estimation* entails finding the value of one or more parameters that make the likelihood of observing the data, considered as a function of the parameters, highest. In many cases, the estimates cannot be determined analytically and one has to resort to iterative algorithms.

This iterative search can be done by updating the current point. Newton-Raphson is a well-known example for this search. Another method is to update the interval in which the maximum has to lie. Here, one takes two points on both sides of the optimum and a third point somewhere in the middle. Depending on the derivative of this point, it replaces one of the other points, and again another point somewhere in the middle is taken. A well-known example is the Bisection method, which lets the third point lie exactly between the two other points.

Both methods have a disadvantage: Newton-Raphson is not guaranteed to converge, whereas interval algorithms do not find a better point in each step. A method which does update the current point in every step and does for certain converge is *Iterative Minorization* (IM).

In this thesis, we focus on the estimation of the parameters of the Rasch Model of psychometrics (Rasch, 1960). For this estimation, we test the performance of the monotone converging algorithms. Not all algorithms we consider in this thesis are new. For example, we discuss the *implicit equations* algorithm that was introduced by Fischer (1974). This algorithm has been used for many years although it was never proven to converge. By showing that this algorithm is based on iterative minorization, we establish its monotone convergence.

For all algorithms, we prove that, under certain conditions, they can be generalized to the class of Exponential Family Models. This Family includes many useful statistical distributions. An important property of the Exponential Family is that

the likelihood functions are log-concave, which is needed for the convergence of the algorithms. This concavity implies there is a unique maximum.

The Rasch model is used to estimate the difficulty parameters of items, for instance the questions on a graduation exam, and the ability parameters of persons, for example, of students. Here, we present how the likelihood function follows from the density function. The probability $f(y_{pi} = 1|\theta_p, \delta_i)$ that person p answers item i correctly lies between 0 and 1, and is determined by a logistic function. This function is monotonically increasing in the ability parameters of the persons, θ_p, and monotonically decreasing in the difficulty parameters of the items, δ_i. The Rasch model assumes that the probability on a correct answer is

$$\frac{\exp(\theta_p - \delta_i)}{1 + \exp(\theta_p - \delta_i)}, \tag{1}$$

and the probability on an incorrect answer is

$$\frac{1}{1 + \exp(\theta_p - \delta_i)}.$$

The probability of giving the right answer is 50% if and only if $\theta_p = \delta_i$, as shown in Figure 1.

Under the assumption that this probability only depends on the parameters, and not on the fact that a person has answered other items right or wrong, the likelihood of the Rasch model can be obtained by multiplying over the probabilities of all observations:

$$\begin{aligned} P(\mathbf{Y}|\boldsymbol{\theta}, \boldsymbol{\delta}) &= \prod_{p=1}^{M} \prod_{i=1}^{N} (I[y_{pi} = 1] f(y_{pi} = 1|\theta_p, \delta_i) + I[y_{pi} = 0] f(y_{pi} = 0|\theta_p, \delta_i)) \\ &= \prod_{p=1}^{M} \prod_{i=1}^{N} (y_{pi} f(y_{pi} = 1|\theta_p, \delta_i) + (1 - y_{pi}) f(y_{pi} = 0|\theta_p, \delta_i)) \\ &= \prod_{p=1}^{M} \prod_{i=1}^{N} \frac{\exp(y_{pi}(\theta_p - \delta_i))}{1 + \exp(\theta_p - \delta_i)}, \end{aligned}$$

where N is the number of items, M the number of persons, $\mathbf{Y}$ the $M \times N$ matrix

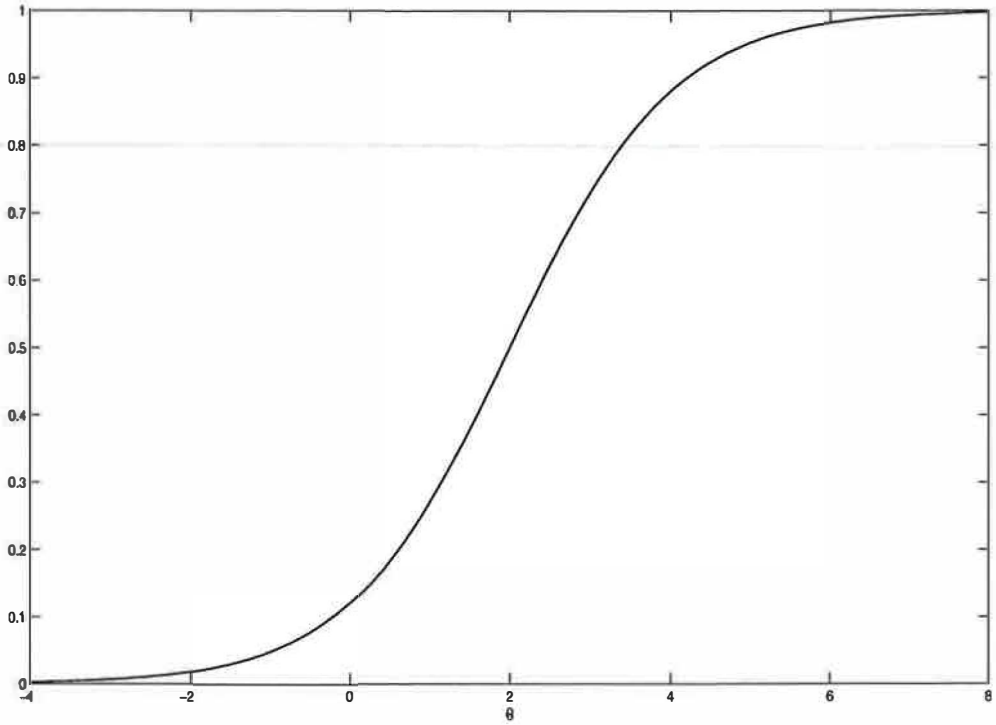

FIGURE 1.

The probability of given the right answer where $\delta_i = 2$

of responses with elements y_{ip} equal being 1 for a correct answer and 0 otherwise, $\boldsymbol{\theta}$ the $M \times 1$ vector of abilities θ_p, and $\boldsymbol{\delta}$ the $N \times 1$ vector with item difficulties δ_i. Note that the last line of this equation covers both the case that the answer is right, as well as the case the answer is wrong.

It is well-known that the parameters of the Rasch model are not identifiable. Specifically, the ability parameters θ_p and the difficulty parameters δ_i can be shifted by the same constant without changing the value of the likelihood function. This problem is resolved here by assuming that the difficulty of the first item is equal to zero. The other difficulty parameters must be interpreted relative to the first item.

Usually, the focus of the analysis are the item parameters δ_i. In that case, we are not interested in the ability parameters θ_p, which can be considered incidental.

Moreover, the number of the ability parameters increases with the sample size. In the presence of an increasing number of incidental parameters it is, in general, not possible to estimate the (structural) difficulty parameters consistently (Kiefer & Wolfowitz, 1956; Neyman & Scott, 1948). This problem can be overcome in one of two ways.

1. Since the Rasch model is an Exponential Family Model we can base our inferences on the distribution of the data $\mathbf{Y}$ conditional on the sufficient statistics for the incidental parameters. This conditional distribution no longer depends on the incidental parameters, and also belongs to the Exponential Family (Lehmann, 1986). This method is called *conditional maximum likelihood* (CML) estimation.
2. If the persons can be conceived as a random sample from a well defined population characterized by an ability distribution G, the structural difficulty parameters can be estimated from the marginal distribution of the data. That is, we integrate the incidental parameters out of the model. Rather than estimating each person's ability, only the parameters of the ability *distribution* need to be estimated. This method is called *marginal maximum likelihood* (MML) estimation. Note that the marginal likelihood does not belong to the Exponential Family.

Under suitable regularity conditions both methods can be shown to lead to consistent estimates of the difficulty parameters.

In this thesis, we concentrate on CML estimation. Because the likelihood function is concave in the difficulty parameters δ_i, we do not have to worry about local maxima. The global optimum of this likelihood cannot be found analytically. In OPLM, a software package used at CITO, currently a combination of the implicit equations algorithm (Fischer, 1974) and Newton-Raphson is used. We compare the performance of these and several other iterative methods to find the global optimum

of the likelihood. We try to find algorithms which are faster than the ones which are currently used with as goal increasing the speed of the software package.

The remainder of this thesis is organized as follows: In Section 2, we present the general idea of iterative minorization, and some practical examples. The conditional log-likelihood function of the Rasch model is derived in section 3. In Sections 4 to 6, we respectively explain how to use minorization algorithms, interval algorithms, and Newton-Raphson. These algorithms are used in a simulation study, which is presented in Section 7. Section 8 contains some final remarks.

2. Iterative Minorization

A function $g(x)$ is said to minorize a target function $f(x)$ if $g(x)$ is smaller than or equal to $f(x)$ for all x, and coincides at least at one point $\hat{x}$. The point where we demand the two functions to coincide is called the supporting point. In this thesis, we assume that the target functions are continuous and twice differentiable. The minorizing condition implies that at $\hat{x}$ the first-order derivatives are equal, and the second-order derivative of $g(x)$ is smaller. In literature, the topic of majorization is discussed (Heiser, 1995; Lange, Hunter, & Yang, 2000). This is used to find the minimum of a function. Since maximizing a function is the same as minimizing minus this function, and bacause minorization is the exact opposite of majorization, the same properties hold (Hunter & Lange, 2004). An important characteristic of minorization is that it is closed under summation, a property that is used later. Because the minorizing function $g(x)$ depends on the supporting point $\hat{x}$, we write $g(x, \hat{x})$. That is, we need a minorizing function $g(x, \hat{x})$ such that:

$$g(x, \hat{x}) \begin{cases} = f(x) & \text{if } x = \hat{x}; \\ \leq f(x) & \text{for all } x \neq \hat{x}. \end{cases}$$

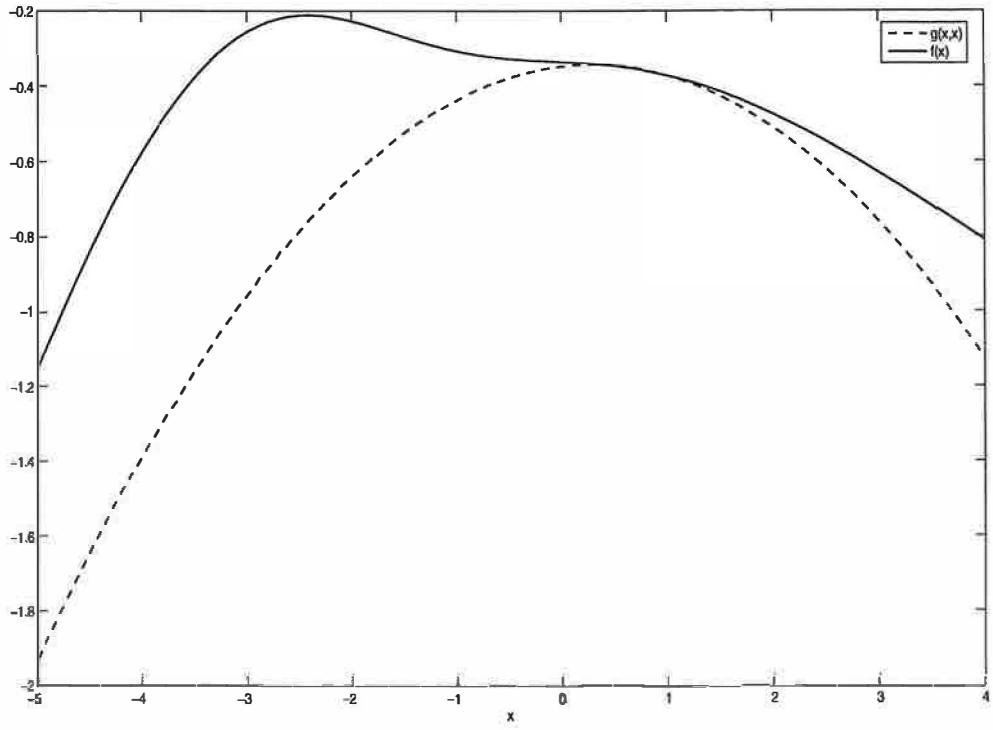

FIGURE 2.

Illustration of minorizing a *difficult* function by a quadratic function

Figure 2 shows how a *difficult* function is minorized by a parabola.

A minorizing function $g(x, \widehat{x})$ is used when the maximum of $f(x)$ cannot easily be found. The maximum of $g(x, \widehat{x})$ can usually be found by setting the first-order derivative equal to zero. The iterative minorization approach for finding the maximum of $f(x)$ rests on iteratively using the Sandwich inequality (De Leeuw, 1994)

$$f(\widehat{x_i}) = g(\widehat{x_i}, \widehat{x_i}) \leq g(\widehat{x_{i+1}}, \widehat{x_i}) \leq f(\widehat{x_{i+1}}),$$

where $g(\widehat{x_{i+1}}, \widehat{x_i})$ is the maximum value of $g(x, \widehat{x_i})$. In the next step, $\widehat{x_{i+1}}$ replaces $\widehat{x_i}$ as supporting point. If the sequence of values of $f(\widehat{x_i})$ is bounded, convergence of this sequence to a maximum is guaranteed. Often, the minorizing function is quadratic, leading to simple updates. However, in this thesis we also investigate the

use of other minorizing functions.

To construct a minorizing function, one has to look closely at the properties of $f(x)$. One has to determine:

1. its domain;
2. its limits;
3. where the function is convex and concave.

Then, one needs to think about the preferable shape of the minorizing function. Generally, linear or quadratic functions are used. Later on, we present a function which is neither linear nor quadratic. At the supporting point, the minorizing function has to have the same value, the same first-order derivative, and a smaller or equal second-order derivative compared to the target function. To satisfy these three conditions, it may help to choose a function with three free parameters a, b, and c, which depends on the supporting point $\widehat{x}$. An example is the quadratic function $g(x, \widehat{x}) = ax^2 + bx + c$. One can replace a and b with functions of c, using the two equality conditions. This setup leaves only the inequality condition of the second-order derivative, and one free parameter c. Choosing a proper c now gives a potentially minorizing function. A picture of the graph gives an indication whether the function actually minorizes, but a formal prove is always needed. In the next two sections, we give some examples to illustrate how minorizing functions can be constructed.

2.1. One supporting point

At first, we give a general solution for minorizing convex functions. Hereafter, we show how the concave function $\ln(x)$ can be minorized by rational functions.

Minorizing convex functions

The following example is more general. Suppose the function $f(x)$ has the following properties:

1. its domain is $[a, b]$;
2. $f(a)$ and $f(b)$ exist;
3. the function is convex.

Because a convex function has a positive second-order derivative for all values, a sufficient minorization is a straight line (Borg & Groenen, 1997; Heiser, 1995). Such a minorization can be found by taking the first-order Taylor approximation at the point $\widehat{x}$:

$$g(x, \widehat{x}) = f(\widehat{x}) + f'(\widehat{x})(x - \widehat{x}). \tag{2}$$

This minorization for example holds for x^2 and $-\ln(x)$, see Figure 3. For $f(x) = x^2$, the Taylor expansion at $\widehat{x}$ is

$$g(x, \widehat{x}) = \widehat{x}^2 + 2\widehat{x}(x - \widehat{x}),$$

and for $f(x) = -\ln(x)$, it is

$$g(x, \widehat{x}) = -\ln(\widehat{x}) - \frac{x - \widehat{x}}{\widehat{x}}.$$

Minorizing $\ln(x)$

Here, we try to minorize the function $f(x) = \ln(x)$. The properties of this function are:

1. it only exists for positive x;
2. $\lim_{x \downarrow 0} f(x) = -\infty$;

   $\lim_{x \to \infty} f(x) = \infty$;

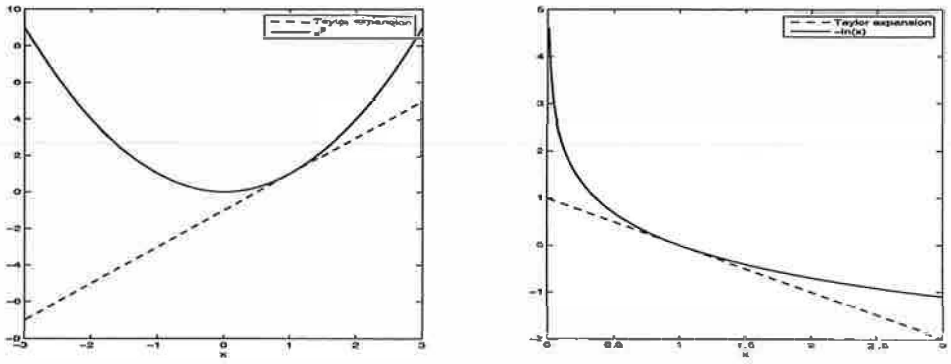

FIGURE 3.
Minorization of x^2 and $-\ln(x)$ by their first-order Taylor expansion.

3. the function is concave on the whole domain.

We have to find a function $g(x, \widehat{x})$ which minorizes $f(x)$ on the interval $(0, \infty)$, and of which $\lim_{x \downarrow 0} g(x, \widehat{x}) = -\infty$. The latter property implies that $g(x, \widehat{x})$ cannot be a polynomial. Therefore, we try two rational functions of the following form:

$$g_1(x, \widehat{x}) = \frac{-a}{x} - bx + c,$$
$$g_2(x, \widehat{x}) = \frac{-a}{x} - bx^2 + c,$$

with $a, b > 0$ to ensure that $g_1(x, \widehat{x})$ and $g_2(x, \widehat{x})$ are smaller in the limits. We must admit that the minorizing functions are quite difficult to optimize, because of the rational part $-a/x$. However, we hope to give the reader a better idea on the construction of minorizing functions.

For $g_1(x, \widehat{x})$ the following conditions have to hold:

$$\frac{-a}{\widehat{x}} - b\widehat{x} + c = \ln(\widehat{x}),$$
$$\frac{a}{\widehat{x}^2} - b = \frac{1}{\widehat{x}},$$
$$\frac{-2a}{\widehat{x}^3} \leq \frac{-1}{\widehat{x}^2}.$$

Expressing a and b as a function of c gives

$$a = 1/2\hat{x}(c - \ln(\hat{x}) + 1),$$
$$b = \frac{c - \ln(\hat{x}) - 1}{2\hat{x}}.$$

The inequality now becomes

$$\frac{-(c - \ln(\hat{x}) + 1)}{\hat{x}^2} \leq \frac{-1}{\hat{x}^2} \Rightarrow c \geq \ln(\hat{x}).$$

To ensure that a and b are positive, c has to be larger than $\ln(\hat{x}) + 1$. If we for instance choose $c = \ln(\hat{x}) + 2$, we end up with

$$g_1(x, \hat{x}) = -\frac{3\hat{x}}{2x} - \frac{x}{2\hat{x}} + \ln(\hat{x}) + 2.$$

Doing the same for $g_2(x, \hat{x})$, gives the following set of equations:

$$\frac{-a}{\hat{x}} - b\hat{x}^2 + c = \ln(\hat{x}),$$
$$\frac{a}{\hat{x}^2} - 2bx = \frac{1}{\hat{x}},$$
$$\frac{-2a}{\hat{x}^3} - 2b \leq \frac{-1}{\hat{x}^2}.$$

Solving these for a, b and c, gives

$$a = 1/3\hat{x}(2c - 2\ln(\hat{x}) + 1),$$
$$b = \frac{c - \ln(\hat{x}) - 1}{3\hat{x}^2},$$
$$c \geq \ln(\hat{x}) + 1/2.$$

Again, because b has to be positive, c has to be larger $\ln(\hat{x}) + 1$. For symmetry, we again choose $\ln(\hat{x}) + 2$. It can be seen that for this value a and b are positive. We now have the minorizing function

$$g_2(x, \hat{x}) = -\frac{3\hat{x}}{x} - \frac{x^2}{3\hat{x}^2} + \ln(\hat{x}) + 2$$

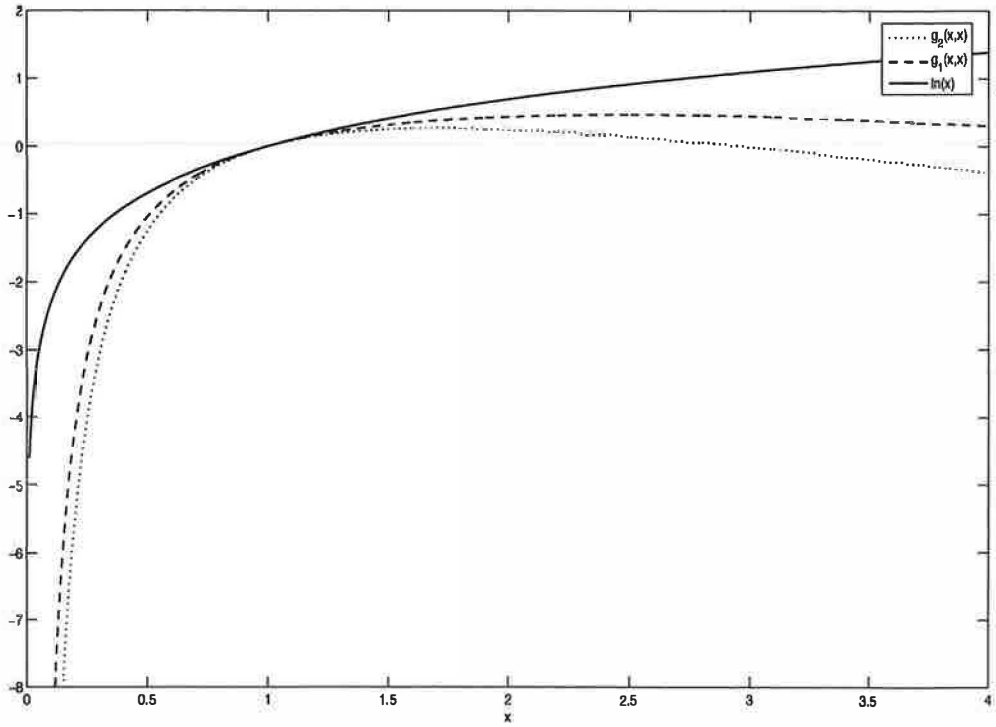

FIGURE 4.

Minorizations of ln(x) with supporting point 1.

The two minorizations are plotted in Figure 4. Both functions have the property that their limits for $x \to 0$, and $x \to \infty$ are equal to $-\infty$. They touch $f(x)$ only at the supporting point $\widehat{x}$.

Now, we look at quadratic minorizing functions with two supporting points. One can use these to minorize concave functions.

2.2. Two supporting points

Here, we show how a concave function $f(x)$ can be minorized by a quadratic function which has two supporting points. Before we give a practical example, we first provide a new theorem stating that such a minorizing function is unique.

Theorem 1. *For a given function, consider minorizing functions of the following*

form:

$$m_{ik}(x, \hat{x}) = a_i x^k + b_i x + c_i,$$

where $\hat{x}$ is the supporting point and k is a fixed integer constant greater than or equal to 2, that is even and finite. There is no value $y \neq \hat{x}$ such that $m_{ik}(y, \hat{x}) = m_{jk}(y, \hat{x})$, for $i \neq j$. Furthermore, for any two such functions one minorizes the other.

Corollary 1. *If the minorization $m_{ik}(x, \hat{x})$ of a certain target function has a second supporting point, this second supporting point is unique, and this is the only function with multiple supporting points for given k.*

Corollary 2. *The unique minorizing function $m_{ik}(x, \hat{x})$ with two supporting points is better than any other minorizing function of the same order k.*

A proof of Theorem 1 and its corollaries is given in the Appendix. From Groenen, Giaquinto, and Kiers (2003) we derive the quadratic minorizing function $g(x, \hat{x})$ for $f(x) = -\ln(1 + \exp(-x))$, with supporting points $\hat{x}$ and $-\hat{x}$:

$$f(x) = -\ln(1 + \exp(-x)) \geq g(x, \hat{x}) = ax^2 + bx + c,$$

with

$$\begin{aligned} a &= \frac{f'(\hat{x}) - f'(-\hat{x})}{4\hat{x}}, \\ b &= -2a\hat{x} + f'(\hat{x}), \\ c &= f(\hat{x}) - a\hat{x}^2 - b\hat{x}, \end{aligned} \tag{3}$$

where $\lim_{x \to 0} a = -1/8$. Groenen et al. (2003) prove that $g(x, \hat{x})$ actually minorizes $f(x)$. Figure 5 plots $f(x)$ and $g(x, \hat{x})$ with supporting points -2 and 2.

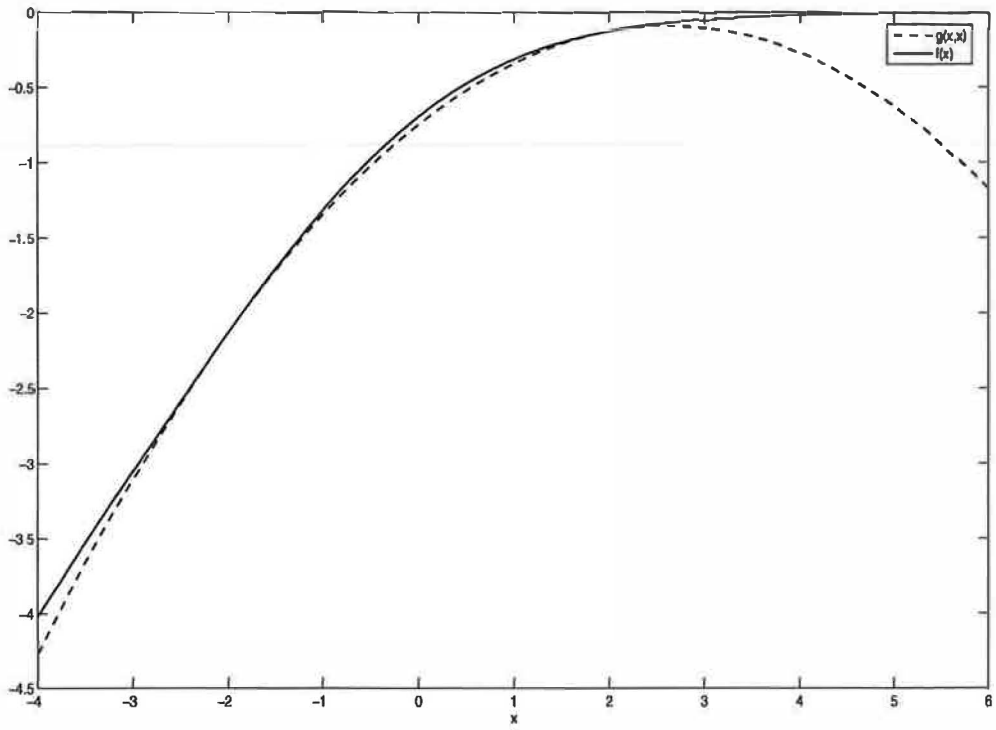

FIGURE 5.

Quadratic minorization of $-\ln(1+\exp(-x))$ with two supporting points.

In this case, $f(x)$ has some interesting properties:

$$\begin{aligned} f(x) &= -\ln(1+\exp(-x)) \\ &= -\ln(1+\exp(-x)) - \ln(\exp(x)) + \ln(\exp(x)) \\ &= -\ln((1+\exp(-x))\exp(x)) + x \\ &= -\ln(1+\exp(x)) + x \\ &= f(-x) + x, \end{aligned} \tag{4}$$

$$f'(x) = -f'(-x) + 1, \tag{5}$$

where (5) follows from (4) via the chain rule. Obviously, for the minorizing function

$g(x,\widehat{x})$ these two properties also have to hold at $\widehat{x}$. Now, suppose $g(x,\widehat{x})$ is a general expression of a quadratic function:

$$g(x,\widehat{x}) = ax^2 + bx + c.$$

Because (4) also has to hold for $g(x,\widehat{x})$ at both supporting points, we obtain

$$ax^2 + bx + c = a(-x)^2 - bx + c + x \Rightarrow 2bx = x \Rightarrow b = 1/2.$$

The latter equation implies that b is independent of $\widehat{x}$! This result can also be derived from a combination of (3) and (5):

$$\begin{aligned} b &= -2a\widehat{x} + f'(\widehat{x}) \\ &= -2\frac{f'(\widehat{x}) - f'(-\widehat{x})}{4} + f'(\widehat{x}) \\ &= 1/2f'(\widehat{x}) + 1/2f'(-\widehat{x}) \\ &= 1/2. \end{aligned}$$

The fact that at the supporting points also (5) has to hold for $g(x,\widehat{x})$, helps to simplify a:

$$a = \frac{2f'(\widehat{x}) - 1}{4\widehat{x}}.$$

In this paper, we consider two iterative minorization algorithms for the Rasch model. Quadratic minorization is used in one of these algorithms. In the next section, we continue our discussion of this model.

3. Rasch Model: The Conditional log-likelihood Function

In the introduction, we have described the Rasch model. In this section, we introduce the conditional log-likelihood function which has to be optimized. We show how this function follows from the general likelihood function. It is convenient

to reparameterize the Rasch model by

$$t_p = \exp(\theta_p),$$

$$b_i = \exp(-\delta_i),$$

$$n_p = \sum_i y_{pi},$$

$$m_i = \sum_p y_{pi},$$

where n_p and m_i are the sufficient statistics for θ_p and δ_i, respectively. Also, we use the following generalization of the Binomial Theorem:

$$\prod_{i=1}^{N}(1 + b_i t) = \sum_{j=0}^{N}\gamma_j(\mathbf{b})t^j,$$

where $\gamma_j(\mathbf{b}) = \sum_{\mathbf{x}:n=j} \prod_i b_i^{x_i}$ are the so-called *elementary symmetric* functions. The latter summation is taken over all response vectors $\mathbf{x}$ which lead to score j. This function is based on writing out the product into a summation by basic mathematics. For notational convenience, we write γ_j for $\gamma_j(\mathbf{b})$. In the Appendix more information is given on these functions. The above functions can now be used to reparameterize the likelihood:

$$\begin{aligned}
P(\mathbf{Y}|\boldsymbol{\theta},\boldsymbol{\delta}) = P(\mathbf{Y}|\mathbf{t},\mathbf{b}) &= \prod_{p=1}^{M}\prod_{i=1}^{N}\frac{\exp(y_{pi}(\theta_p - \delta_i))}{1+\exp(\theta_p - \delta_i)} \\
&= \prod_{p=1}^{M}\prod_{i=1}^{N}\frac{\exp(\theta_p y_{pi})\exp(-\delta_i y_{pi})}{1+\exp(\theta_p)\exp(-\delta_i)} \\
&= \prod_{p=1}^{M}\prod_{i=1}^{N}\frac{t_p^{y_{pi}} b_i^{y_{pi}}}{1+t_p b_i} \\
&= \prod_{p=1}^{M}\frac{t_p^{n_p}\prod_{i=1}^{N} b_i^{y_{pi}}}{\prod_{i=1}^{N}(1+b_i t_p)} \\
&= \prod_{p=1}^{M}\frac{t_p^{n_p}\prod_{i=1}^{N} b_i^{y_{pi}}}{\sum_{j=0}^{N}\gamma_j t_p^j}.
\end{aligned}$$

We are interested in the likelihood conditional on the $Mx1$ vector of sum scores $\mathbf{n}$. This conditional likelihood is defined as

$$P(\mathbf{m}|\mathbf{n},\mathbf{b}) = \frac{P(\mathbf{m}, \sum_i \mathbf{y} = \mathbf{n}|\mathbf{t},\mathbf{b})}{P(\sum_i \mathbf{y} = \mathbf{n}|\mathbf{t},\mathbf{b})}.$$

The nominator is the likelihood, whereas the denominator is equal to

$$\begin{aligned} P(\sum_i \mathbf{y} = \mathbf{n}|\mathbf{t},\mathbf{b}) &= \prod_{p=1}^{M} \sum_{\mathbf{y}:\sum_k y_{pk}=n_p} \frac{t_p^{n_p} \prod_{i=1}^{N} b_i^{y_{pi}}}{\sum_{j=0}^{N} \gamma_j t_p^j} \\ &= \prod_{p=1}^{M} \frac{t_p^{n_p}}{\sum_{j=0}^{N} \gamma_j t_p^{n_p}} \sum_{\mathbf{y}:\sum_k y_{pk}=n_p} \prod_{i=1}^{N} b_i^{y_{pi}}. \end{aligned}$$

The conditional likelihood can now be obtained by

$$\begin{aligned} \frac{P(\mathbf{m}, \sum_i \mathbf{y} = \mathbf{n}|\mathbf{t},\mathbf{b})}{P(\sum_i \mathbf{y} = \mathbf{n}|\mathbf{t},\mathbf{b})} &= \prod_{p=1}^{M} \prod_{i=1}^{N} b_i^{y_{pi}} \left(\sum_{\mathbf{y}:\sum_k y_{pk}=n_p} \prod_{i=1}^{N} b_i^{y_{pi}} \right)^{-1} \\ &= \frac{\prod_{i=1}^{N} b_i^{m_i}}{\prod_{p=1}^{M} \gamma_{n_p}}, \end{aligned}$$

where the last equality follows from the definition of the elementary symmetric functions. Since the sum scores $\mathbf{n}$ are sufficient for the ability parameters $\mathbf{t}$ this conditional distribution no longer depends on $\mathbf{t}$. Taking the logarithm gives the conditional log-likelihood function (CLLF)

$$\begin{aligned} l_{\mathbf{m}|\mathbf{n}}(\mathbf{b}) &= \sum_i m_i \ln(b_i) - \sum_p \ln(\gamma_{n_p}) \\ &= \sum_i m_i \ln(b_i) - \sum_p \ln(\gamma_{n_p}^{(j)} + b_j \gamma_{n_p-1}^{(j)}), \end{aligned}$$

where $\gamma_n^{(j)}$ is used to denote the n-th elementary symmetric function of $\mathbf{b}$ without the j-th element (see the Appendix). The previously used parameterization has the restriction that the b_i's always have to be positive. During an algorithm, they can become negative, and we find no solution. Therefore, the CLLF usually is reparam-

eterized, where b_i is replaced by $\exp(-\delta_i)$:

$$\begin{aligned} l_{\mathbf{m}|\mathbf{n}}(\boldsymbol{\delta}) &= -\sum_i m_i\delta_i - \sum_p \ln(\gamma_{n_p}) \\ &= -\sum_i m_i\delta_i - \sum_p \ln(\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)}). \end{aligned} \tag{6}$$

The advantage of (6) is that it exists for every value of δ_j. Because $\gamma_n^{(j)}$ is always positive, this function is concave in every δ_j, and has no local maxima. We prove this in Section 4.5 by showing that all Exponential Family Models have a concave log-likelihood. In the next section, we introduce two iterative minorization algorithms, starting with an estimation procedure that is currently used in software.

4. IM Algorithms for the Rasch Model

In the Rasch model, maximization cannot be done analytically by derivation, because the right-hand side of the CLLF is intractable. In this section, we focus on methods which use iterative minorization to find the maximum of this concave function. One can choose to use a simple minorization, which for instance uses a Taylor expansion of the logarithmic part of the log-likelihood function. Another idea is to use a more difficult minorization, which uses two supporting points. This minorization probably costs more calculation time per iteration, but this function is expected to lie closer to the log-likelihood function.

In OPLM, a software package used at CITO, currently a combination of the implicit equations algorithm (Fischer, 1974) and Newton-Raphson is used. The proof of the (monotone) convergence of the implicit equations algorithm has not been given until now. We present this proof by showing that this algorithm is a form of iterative minorization. Secondly, we present a minorizing function which uses two supporting points. Finally, we illustrate how both ideas can be generalized to the Exponential Family.

4.1. One supporting point: Implicit equations

The terms $-\ln(\gamma_{n_p}^{(j)} + b_j\gamma_{n_p-1}^{(j)})$ of the CLLF are convex functions in b_j ($\gamma_n^{(j)}$ does not depend on b_j). Because one wants to minorize the CLLF, one can opt for minorizing these terms. This minorization can be done by taking the first-order Taylor expansion, see Section 2.2. We obtain

$$-\ln(\gamma_{n_p}) \geq -\left(\ln(\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}) + \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}}(b_j - \widehat{b_j})\right).$$

By substituting the right-hand side for the left hand side in the CLLF, we obtain the minorizing function $M_{\mathbf{m}|\mathbf{n}}(b_j, \mathbf{b})$, see Figure 6:

$$M_{\mathbf{m}|\mathbf{n}}(b_j, \mathbf{b}) =$$
$$\sum_i m_i \ln(b_i) - \sum_p \left(\ln(\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}) + \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}}(b_j - \widehat{b_j})\right) \leq l_{\mathbf{m}|\mathbf{n}}(\mathbf{b}).$$

The CLLF and the minorizing function coincide at $b_j = \widehat{b_j}$, as they should. If we differentiate $M_{\mathbf{m}|\mathbf{n}}(b_j, \mathbf{b})$ with respect to b_j we obtain

$$\frac{m_j}{b_j} - \sum_p \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}}. \tag{7}$$

Hence, the maximum of $M_{\mathbf{m}|\mathbf{n}}(b_j, \mathbf{b})$ is found at

$$b_j = m_j \left(\sum_p \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}}\right)^{-1}, \tag{8}$$

which is called an implicit equation (b_j is a function of itself). The new b_j can now be used to update b_{j+1}, etcetera, until convergence.

Characteristics of implicit equations

The implicit equations algorithm updates one parameter at every step. Within each step, a minorizing function is updated, hence the CLLF increases until convergence. Suppose we keep the other parameters constant, and only update b_j. Let the

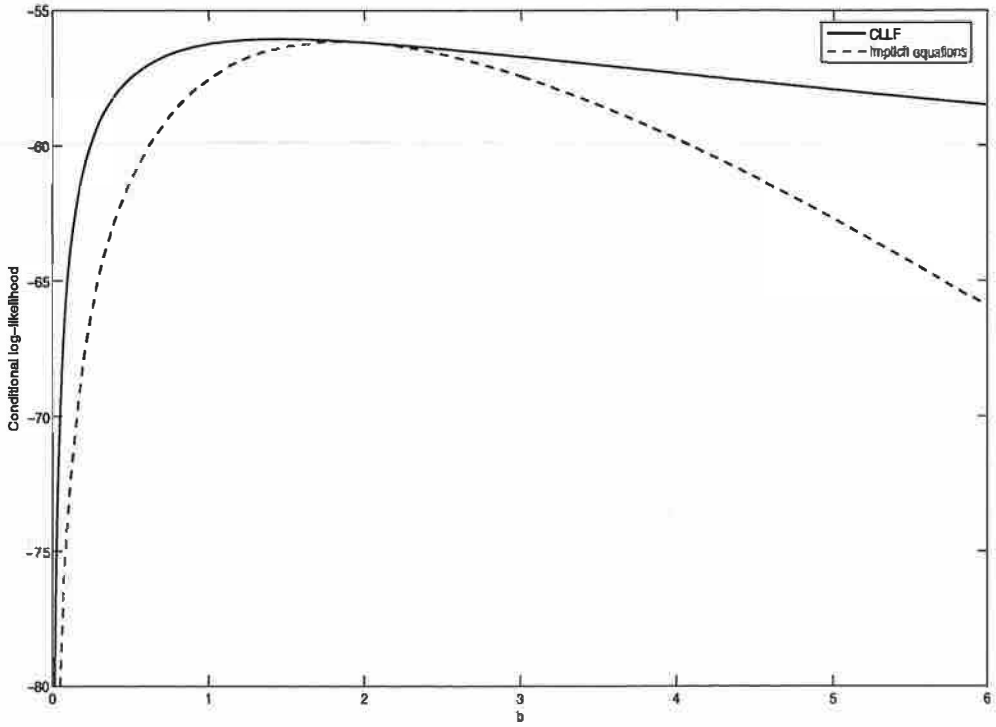

FIGURE 6.

Minorization of the CLLF by Implicit equations with supporting point 2.

top of the function for b_j be $b_{j,opt}$, and the value of b_j found at the i-th iteration $\widehat{b_{j,i}}$. Now, we obtain for every step in this dimension the following theorem:

Theorem 2. *$\widehat{b_{j,i}}$ lies between the old value $\widehat{b_{j,i-1}}$ and the top of the function $b_{j,opt}$.*

Proof. The CLLF is monotonically increasing for $0 < b_j < b_{j,opt}$ and monotonically decreasing for $b_{j,opt} < b_j < \infty$. Suppose that $\widehat{b_{j,i-1}} < b_{j,opt}$, then the slope for $\widehat{b_{j,i-1}}$ is positive:

$$\frac{m_j}{\widehat{b_{j,i-1}}} - \sum_p \frac{\gamma^{(j)}_{n_p-1}}{\gamma^{(j)}_{n_p} + \widehat{b_{j,i-1}}\gamma^{(j)}_{n_p-1}} > 0.$$

This inequality implies that

$$\widehat{b_{j,i}} = m_j \left(\sum_p \frac{\gamma^{(j)}_{n_p-1}}{\gamma^{(j)}_{n_p} + \widehat{b_{j,i-1}}\gamma^{(j)}_{n_p-1}} \right)^{-1} > \widehat{b_{j,i-1}}. \tag{9}$$

The slope of $\widehat{b_{j,i}}$ can be found after filling in the new value:

$$\frac{m_j}{\widehat{b_{j,i}}} - \sum_p \frac{\gamma^{(j)}_{n_p-1}}{\gamma^{(j)}_{n_p} + \widehat{b_{j,i}}\gamma^{(j)}_{n_p-1}} =$$

$$\sum_p \frac{\gamma^{(j)}_{n_p-1}}{\gamma^{(j)}_{n_p} + \widehat{b_{j,i-1}}\gamma^{(j)}_{n_p-1}} - \sum_p \frac{\gamma^{(j)}_{n_p-1}}{\gamma^{(j)}_{n_p} + \widehat{b_{j,i}}\gamma^{(j)}_{n_p-1}} > 0,$$

where the latter inequality follows from inequality (9). Therefore, the signs of the slopes of the new value $\widehat{b_{j,i}}$ and the old value $\widehat{b_{j,i-1}}$ are the same. Combining the fact that the slopes have the same sign, and that $\widehat{b_{j,i}} > \widehat{b_{j,i-1}}$ proves that $\widehat{b_{j,i}} < \widehat{b_j} < b_{j,opt}$. The opposite holds when the slope for $\widehat{b_{j,i-1}}$ is negative. This completes the proof. □

If this procedure is repeated for a single parameter, ceteris paribus, it converges to $b_{j,opt}$. After updating other parameters, this $b_{j,opt}$ also changes. In the implicit equations algorithm, only one updating step per parameter is used. However, it might be advantageous to take more steps, while keeping the other parameters constant. In this way, we gain insight in a possible convergence rate, which can help creating a faster algorithm.

Convergence rate

Here, we are looking for a way to increase the convergence rate of the implicit equations algorithm by investigating the sequence of values of $\widehat{b_{j,i}}$, ceteris paribus. If the differences between the sequential values become smaller, it might be possible to accelerate the procedure to get closer to $b_{j,opt}$, while needing only a few values. We show that such a method can be used by considering n persons and only two items. This example serves to illustrate how we can use the convergence rate to compute the optimum. Note that persons with all items wrong or all items correct contribute nothing to the estimation and are ignored. In both cases, there is only one response pattern that leads to the observed number correct score. Hence, conditional probabilities are either zero or one, which means that they disappear from

the (log)likelihood (see also Fischer, 1974).

For two items, CML estimation entails maximizing the CLLF

$$l_{\mathbf{m}|\mathbf{n}}(b_2) = m_2 \ln(b_2) - r_1 \ln(b_2 + 1),$$

where r_1 denotes the number of persons with one item correct; either item 1 or item 2. These persons constitute all data used for estimation. For notational convenience, the symbol π is used to denote m_2/r_1, the proportion of the r_1 persons with correct answers to item 2.

The update in the implicit equations algorithm is $b_{2,k+1} = \pi(b_{2,k} + 1)$. The first three values can be found, using

$$\begin{aligned}
b_{2,1} &= \pi(1 + b_{2,0}), \\
b_{2,2} &= \pi(1 + b_{2,1}) \\
&= \pi\left(1 + [1 + b_{2,0}]\,\pi\right) \\
&= \pi + \pi^2 + b_{2,0}\pi^2, \\
b_{2,3} &= \pi(1 + b_{2,2}) \\
&= \pi\left(1 + \left[\pi + \pi^2 + b_{2,0}\pi^2\right]\right) \\
&= \pi + \pi^2 + \pi^3 + b_{2,0}\pi^3,
\end{aligned}$$

which can be generalized to

$$\begin{aligned}
b_{2,k} = (1 + b_{2,k-1})\pi &= \left(1 + \left[\sum_{h=1}^{k-1} \pi^h + b_{2,0}\pi^{k-1}\right]\right)\pi \\
&= \sum_{h=1}^{k} \pi^h + b_{2,0}\pi^k.
\end{aligned}$$

We know that for $|z| < 1$

$$\lim_{n\to\infty} s_n = z^1 + z^2 + z^3 + \cdots + z^n = \frac{z}{1 - z}.$$

It follows that the iterates $b_{2,k}$ constitute approximately a geometric series, of which the latter term converges to zero. Since $\pi < 1$, the series converges to

$$\lim_{k\to\infty} b_{2,k} = b_{2,opt} = \frac{\pi}{1-\pi} = \frac{m_2}{r_1 - m_2}.$$

Thus, in this case we may determine the estimates without iterations. Furthermore, if we look at successive iterates, we see that in general

$$|b_{2,k+1} - b_{2,k}| = \pi |b_{2,k} - b_{2,k-1}|,$$

and distances between subsequent iterates become smaller during iterations. Finally, the speed of convergence is seen to depend on m_2/r_1. That is, convergence is slower if the first item is relatively easy so that m_2/r_1 is close to one. We illustrate that this theory also holds for multiple items in the simulation study at the end of this thesis.

Naturally, the problem with two items has an explicit solution and we would not consider an iterative procedure. In case we consider a number of items greater than 2, there is no explicit solution, and it might be possible to use the convergence rate. We know that, in general, a series that is both bounded and monotonically increasing or monotonically decreasing, converges to a number, not further than the bound. In this case the bound is $b_{j,opt}$ and the values decrease or increase monotonically. However, the absolute value of the step $|b_{j,i+1} - b_{j,i}|$ is not necessarily smaller than the absolute value of the previous step, $|b_{j,i} - b_{j,i-1}|$. Some preliminary experimentation shows that the steps usually become smaller, but not always. We prove our last statement by a counter example.

Assume we have three items and three persons. Given are m_1, n_0, n_1, n_2, n_3, b_2 and b_3. We start at the point b_{11}. Assuming that there are no persons who answered

all questions right or wrong, the new values b_{12} and b_{13} can be computed as

$$b_{12} = m_1 \left(n_1 \left(\frac{\gamma_0^{(j)}}{\gamma_1^{(j)} + b_{11}\gamma_0^{(j)}} \right) + n_2 \left(\frac{\gamma_1^{(j)}}{\gamma_2^{(j)} + b_{11}\gamma_1^{(j)}} \right) \right)^{-1}$$
$$= m_1 \left(\frac{n_1}{b_{11} + b_2 + b_3} + \frac{n_2(b_2 + b_3)}{b_{11}b_2 + b_{11}b_3 + b_2b_3} \right)^{-1},$$
$$b_{13} = m_1 \left(\frac{n_1}{b_{12} + b_2 + b_3} + \frac{n_2(b_2 + b_3)}{b_{12}b_2 + b_{12}b_3 + b_2b_3} \right)^{-1}.$$

If we take $m_1 = 2, n_1 = 2, n_2 = 1, b_2 = 0.05, b_3 = 1.4, b_{11} = 0.05$, then b_{12} is 0.1738, and b_{13} is 0.3487. In this case, $|b_{12} - b_{11}| = 0.1238$ and $|b_{13} - b_{12}| = 0.1749$ showing that the steps get larger. We can conclude that although convergence to a certain point is guaranteed, not every step towards this point has to be smaller than the previous one.

Aitken's Convergence Acceleration Method

In case the steps do get smaller, the sequence can be accelerated. Several methods exist which make use of the convergence rate to find the point of convergence x_c. A well-known method is the Aitken's Convergence Acceleration Method (Aitken, 1926). For the implicit equations algorithm, one needs three consecutive points to estimate the one-dimensional optimum $x_c = b_{j,opt}$.

Suppose $x_0 = \widehat{b_{j,0}}$, $x_1 = \widehat{b_{j,1}}$, and $x_2 = \widehat{b_{j,2}}$ are known. The first step is $x_1 - x_0$, the second step $x_2 - x_1$. The growth rate of these steps is

$$\frac{x_2 - x_1}{x_1 - x_0}.$$

Assuming the sequence is a geometric, x_c is equal to

$$x_2 + (x_2 - x_1) \lim_{n \to \infty} \sum_{i=1}^{n} \left(\frac{x_2 - x_1}{x_1 - x_0} \right)^i.$$

Hence, if $|(x_2 - x_1)/(x_1 - x_0)| < 1$, then the point of convergence can be computed

as follows:

$$\begin{aligned} x_c &= x_2 + (x_2 - x_1) \lim_{n\to\infty} \sum_{i=1}^{n} \left(\frac{x_2 - x_1}{x_1 - x_0}\right)^i \\ &= x_2 + (x_2 - x_1) \frac{x_2 - x_1}{x_1 - x_0} \left(1 - \frac{x_2 - x_1}{x_1 - x_0}\right)^{-1} \\ &= x_2 + \frac{(x_2 - x_1)^2}{(x_1 - x_0) - (x_2 - x_1)} \\ &= x_2 + \frac{(x_2 - x_1)^2}{2x_1 - x_0 - x_2}. \end{aligned}$$

For the Rasch model, $b_{j,opt}$ can be estimated in the same way as x_c by using the sequential values $\widehat{b_{j,0}}$, $\widehat{b_{j,1}}$, and $\widehat{b_{j,2}}$. Note that in case the steps become larger, this acceleration method does not work.

4.2. Multiple supporting points: Quadratic minorization

In this section, we present a new minorizing function for the CLLF which has two supporting points by again minorizing the right part of the CLLF. The goal of minorization with multiple supporting points is to find a function which is closer to the target function than a minorization which is obtained with only one supporting point.

In Section 2.3, $f(x) = -\ln(1 + \exp(-x))$ is minorized by a quadratic function. This quadratic function has two supporting points, which lie symmetrically around zero, and is lower than $f(x)$ for all other x. Now to obtain a term of the CLLF, two factors are added to parts of $f(x)$. For notational convenience, we write $\gamma_0 = \gamma_{n_p}^{(j)}$, $\gamma_1 = \gamma_{n_{p-1}}^{(j)}$ and $x = \delta_j$:

$$k(x) = -\ln(\gamma_0 + \gamma_1 \exp(-x)), \tag{10}$$

which can be rewritten as

$$\begin{aligned} k(x) &= -\ln(1 + \frac{\gamma_1}{\gamma_0}\exp(-x)) - \ln(\gamma_0) \\ &= -\ln(1 + \exp(\ln\left(\frac{\gamma_1}{\gamma_0}\right) - x)) - \ln(\gamma_0). \end{aligned}$$

Here, it becomes clear that these factors shift $f(x)$ on the x-axis to the right with the term $\ln(\gamma_1/\gamma_0)$, and on the y-axis down with the term $\ln(\gamma_0)$. The latter shift is not important for finding the parameters, because this only affects the constant c. Correcting for the horizontal shift is done by taking the second supporting point on the other side of a *mirror point*. This mirror point x_m is equal to

$$x_m = \ln\left(\frac{\gamma_1}{\gamma_0}\right).$$

If $\gamma_0 = \gamma_1$, this mirror point is equal to zero, and we are back at $f(x)$ minus the constant $\ln(\gamma_0)$. The parameters of the minorizing function change accordingly:

$$\begin{aligned} \overline{x_2} &= 2\ln\left(\frac{\gamma_1}{\gamma_0}\right) - \widehat{x} \\ a &= \frac{k'(\overline{x_2}) - k'(\widehat{x})}{4(x_m - \widehat{x})} = \frac{k'(\overline{x_2}) - k'(\widehat{x})}{2(\overline{x_2} - \widehat{x})}, \\ b &= k'(\widehat{x}) - 2a\widehat{x}, \\ c &= k(\widehat{x}) - a\widehat{x}^2 - b\widehat{x}. \end{aligned}$$

We use these parameters to create the following minorization:

$$\begin{aligned} l_{\mathbf{m}|\mathbf{n}}(\boldsymbol{\delta}) &= -\sum_i m_i\delta_i - \sum_p \ln(\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)}) \\ &\geq -\sum_i m_i\delta_i + \sum_p (a_p\delta_j^2 + b_p\delta_j + c_p). \end{aligned}$$

The terms on the right-hand side are all minorized with two supporting points. However, unless all $\gamma_{n_p-1}^{(j)}/\gamma_{n_p}^{(j)}$ are equal, the total minorization only has one supporting point $(\widehat{\delta_j})$. Setting the first-order derivative of the minorizing function equal to zero,

gives the new point δ_j:

$$-m_j + \sum_p (2a_p\delta_j + b_p) = 0$$

$$\delta_j = \frac{m_j - \sum_p b_p}{\sum_p 2a_p}$$

The latter is also an implicit equation, because a_p and b_p depend on δ_j for every p. This algorithm is called below the quadratic minorization algorithm. We do not further investigate the behavior of this algorithm, but compare its fit with that of the implicit equations algorithm.

4.3. Comparison between both minorizations

We compare the two minorization algorithms in two ways. First, we consider the global fit of both minorizations. Secondly, we compare the second-order derivatives at the supporting point. We know that at the supporting point, both algorithms have the same value, as well as the same derivative for every term. It is obvious that each term the quadratic minorization function is better at its second supporting point, because then it is equal to the term of the CLLF. To compare both algorithms, we plot them in Figure 7 for constant γ's with four different supporting points. We

make the plot after reparameterizing the implicit equations minorization as

$$
\begin{aligned}
v(b_j, \mathbf{b}) &= -\ln(\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}) - \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \widehat{b_j}\gamma_{n_p-1}^{(j)}}(b_j - \widehat{b_j}) = \\
v(\delta_j, \boldsymbol{\delta}) &= -\ln(\gamma_{n_p}^{(j)} + \exp(-\widehat{\delta_j})\gamma_{n_p-1}^{(j)}) \\
&\quad - \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \exp(-\widehat{\delta_j})\gamma_{n_p-1}^{(j)}}(\exp(-\delta_j) - \exp(-\widehat{\delta_j})) \\
&= -\ln(\gamma_{n_p}^{(j)} + \exp(-\widehat{\delta_j})\gamma_{n_p-1}^{(j)}) + \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \exp(-\widehat{\delta_j})\gamma_{n_p-1}^{(j)}}\exp(-\widehat{\delta_j}) \\
&\quad - \frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)} + \exp(-\widehat{\delta_j})\gamma_{n_p-1}^{(j)}}\exp(-\delta_j)) \\
&= r + s\exp(-\delta_j),
\end{aligned}
$$

where r and s are constant with respect to δ_j, depending on the γ's and the supporting point.

In Figure 7 we see that independently of the supporting point, the implicit equations minorization is better than the quadratic minorization if δ_j gets very large, and worse if δ_j gets very small. In these graphs, it holds that if the supporting point $\widehat{\delta_j}$ is positive, the implicit equations minorization seems better than the quadratic minorization around this supporting point. In case the supporting point is negative, the quadratic minorization seems better in this area.

To see if one minorization dominates the other close to the supporting point, we compare the second-order derivatives of these terms. The dominating function has the largest second-order derivative at the supporting point. We derive the second-order derivatives of the implicit equations minorization, where we use the fact that its value and the value of its first-order derivative are equal to the values of the

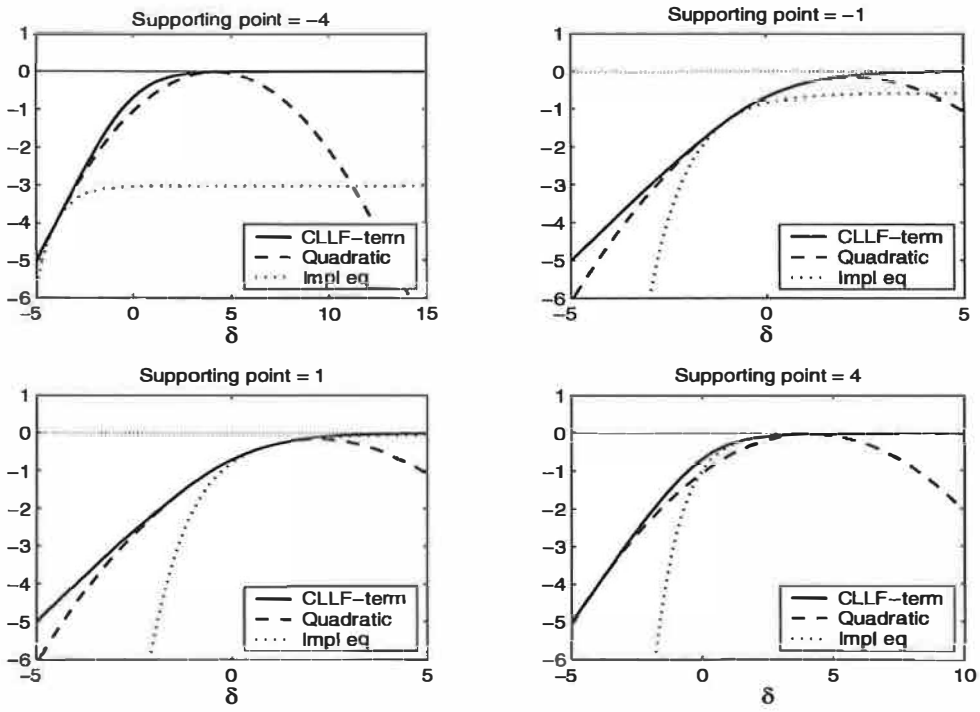

FIGURE 7.

Comparison between two terms of both minorizations.

quadratic minorization.

$$v(\widehat{\delta_j}, \boldsymbol{\delta}) = r + s\exp(-\widehat{\delta_j}) =$$

$$u(\widehat{\delta_j}, \boldsymbol{\delta}) = a\widehat{\delta_j}^2 + b\widehat{\delta_j} + c,$$

$$v'(\widehat{\delta_j}, \boldsymbol{\delta}) = -s\exp(-\widehat{\delta_j}) =$$

$$u'(\widehat{\delta_j}, \boldsymbol{\delta}) = 2a\widehat{\delta_j} + b,$$

$$v''(\widehat{\delta_j}, \boldsymbol{\delta}) = s\exp(-\widehat{\delta_j}) = -2a\widehat{\delta_j} - b,$$

where the last equality follows from the equal first-order derivatives. We want to

know the sign of the difference of the second-order derivatives at $\widehat{\delta_j}$:

$$\begin{aligned} u''(\widehat{\delta_j}, \boldsymbol{\delta}) - v''(\widehat{\delta_j}, \boldsymbol{\delta}) &= 2a + 2a\widehat{\delta_j} + b \\ &= 2a + 2a\widehat{\delta_j} + k'(\widehat{\delta_j}) - 2a\widehat{\delta_j} \\ &= 2a + k'(\widehat{\delta_j}) \\ &= 2\frac{k'(\overline{\delta_{j2}}) - k'(\widehat{\delta_j})}{2(\overline{\delta_{j2}} - \widehat{\delta_j})} + k'(\widehat{\delta_j}), \end{aligned}$$

where $k'(\widehat{\delta_j})$ comes from the previous section. $k'(\widehat{\delta_j})$ is always positive, whereas a is always negative. The difference only depends on two supporting points $\overline{\delta_{j2}}$, and their derivatives. To get a better insight in the sign of the difference, we rewrite $\overline{\delta_{j2}}$, $k'(\widehat{\delta_j})$, and $k'(\overline{\delta_{j2}})$, where we again write γ_0, and γ_1 instead of γ_{n_p} and γ_{n_p-1} respectively:

$$\begin{aligned} \overline{\delta_{j2}} &= 2\ln\left(\frac{\gamma_1}{\gamma_0}\right) - \widehat{\delta_j} \\ k'(\widehat{\delta_j}) &= \frac{\gamma_1 \exp(-\widehat{\delta_j})}{\gamma_0 + \gamma_1 \exp(-\widehat{\delta_j})} \\ &= \frac{\exp(-\widehat{\delta_j})}{\frac{\gamma_0}{\gamma_1} + \exp(-\widehat{\delta_j})} \\ k'(\overline{\delta_{j2}}) &= \frac{\gamma_1 \exp(-\overline{\delta_{j2}})}{\gamma_0 + \gamma_1 \exp(-\overline{\delta_{j2}})} \\ &= \frac{\gamma_1 \exp(-2\ln\left(\frac{\gamma_1}{\gamma_0}\right) - \widehat{\delta_j})}{\gamma_0 + \gamma_1 \exp(-2\ln\left(\frac{\gamma_1}{\gamma_0}\right) - \widehat{\delta_j})} \\ &= \frac{\exp(-2\ln\left(\frac{\gamma_1}{\gamma_0}\right) - \widehat{\delta_j})}{\frac{\gamma_0}{\gamma_1} + \exp(-2\ln\left(\frac{\gamma_1}{\gamma_0}\right) - \widehat{\delta_j})} \end{aligned}$$

These terms all depend on only $\widehat{\delta_j}$ and the ratio of γ_0 and γ_1. Therefore, we expect the same for the difference between the second-order derivatives. We show this in Figure 8.

This figure confirms our expectations that the difference between the second-order derivatives depends on the ratio of γ_0 and γ_1, and the implicit equations

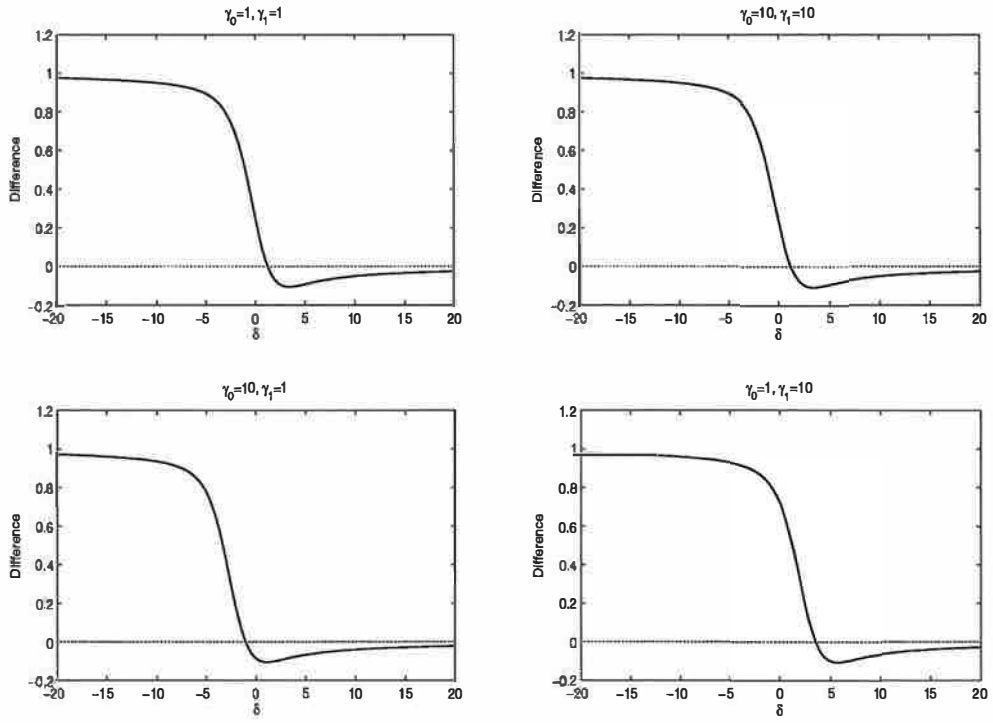

FIGURE 8.

Difference between the second-order derivatives of two terms of both minorizations.

minorization is better around the supporting point than the quadratic minorization if δ_j is larger than some value. This value seems to be slightly bigger than zero, dependent on the ratio. When the ratio γ_1/γ_0 gets larger, the graph shifts to the right. This shift works in favor of the quadratic minorization, which then becomes better on a bigger interval. Note that the advantage of the quadratic minorization is usually larger than the advantage of the implicit equations minorization.

4.4. Generalization to Exponential Family Models

The two minorization techniques which are used for the Rasch model can also be applied in the estimation of the parameters of other Exponential Family Models with finite, discrete data. We show this after a short introduction for this class of

models, and show that its likelihood functions have a comparable shape to the Rasch model. This is hardly surprising, since the Rasch model belongs to this family.

The likelihood of Exponential Family Models with discrete data has the following form:

$$P(\mathbf{x}|\boldsymbol{\phi}) = \frac{b(\mathbf{x})\exp(\boldsymbol{\phi}^T\mathbf{d}(\mathbf{x}))}{a(\boldsymbol{\phi})},$$

where $\mathbf{x}$ symbolizes the data and $\boldsymbol{\phi}$ a vector of parameters, $b(\mathbf{x})$ is a function of the data, $\mathbf{d}(\mathbf{x})$ a vector of the sufficient statistics, and $a(\boldsymbol{\phi})$ a scaling function. A property of a density is that the probability of being somewhere in the whole domain is equal to one:

$$\sum_{\boldsymbol{x}} \mathbf{x} \frac{b(\mathbf{x})\exp(\boldsymbol{\phi}^T\mathbf{d}(\mathbf{x}))}{a(\boldsymbol{\phi})} = 1.$$

This restriction leads to the definition of $a(\boldsymbol{\phi})$:

$$a(\boldsymbol{\phi}) = \sum_{\mathbf{x}} b(\mathbf{x})\exp(\boldsymbol{\phi}^T\mathbf{d}(\mathbf{x})),$$

with first-order derivative

$$\frac{\partial a(\boldsymbol{\phi})}{\partial \boldsymbol{\phi}_i} = \sum_{\mathbf{x}} b(\mathbf{x}) d_i(\mathbf{x})\exp(\boldsymbol{\phi}^T\mathbf{d}(\mathbf{x})),$$

where $d_i(\mathbf{x})$ denotes the i-th element of $\mathbf{d}(\mathbf{x})$. We can compare this with the Rasch model. Here, $b(\mathbf{x})$ is equal to 1, $\boldsymbol{\phi}$ is the vector $\begin{bmatrix} \boldsymbol{\theta} \\ -\boldsymbol{\delta} \end{bmatrix}$, $\mathbf{d}(\mathbf{x})$ is $\begin{bmatrix} \mathbf{n} \\ \mathbf{m} \end{bmatrix}$, and the denominator also sums over all possible values. To find the maximum of the likelihood, one usually takes the logarithm (one-one transformation), and differentiates this function to $\boldsymbol{\phi}_i$. The maximum of the (log)likelihood function is reached when $d_i(\mathbf{x})$ equals its

expected value (Andersen, 1980):

$$\begin{aligned}
\ln P(\mathbf{x}|\boldsymbol{\phi}) &= \ln b(\mathbf{x}) + \boldsymbol{\phi}^T \mathbf{d}(\mathbf{x}) - \ln a(\boldsymbol{\phi}), \\
\frac{\partial}{\partial \boldsymbol{\phi}_i} \ln P(\mathbf{x}|\boldsymbol{\phi}) &= d_i(\mathbf{x}) - E[d_i(X)], \\
\frac{\partial^2}{\partial \boldsymbol{\phi}_i \partial \boldsymbol{\phi}_i} \ln P(\mathbf{x}|\boldsymbol{\phi}) &= -(E[d_i^2(X)] - E[d_i(X)]^2) \\
&= -V[d_i(X)].
\end{aligned}$$

The second-order derivative is equal to minus the variance of $\mathbf{d}_i(\mathbf{x})$, which is positive by definition. Therefore, the log-likelihood function is concave, and the likelihood itself log-concave. This log-likelihood consists of an *easy* part, and a *difficult* logarithmic part, $-\ln a(\boldsymbol{\phi})$. We show that for finite data, one can minorize the latter part by rewriting this part with ϕ_i as free parameter.

$$\begin{aligned}
-\ln a(\boldsymbol{\phi}) &= -\ln(\sum_{\mathbf{x}} b(\mathbf{x}) \exp(\sum_j \phi_j d_j(\mathbf{x}))) \\
&= -\ln(\sum_{\mathbf{x}} b(\mathbf{x}) \exp(\sum_{j \neq i} \phi_j d_j(\mathbf{x})) \exp(\phi_i d_i(\mathbf{x}))) \\
&= -\ln(\sum_d \sum_{\mathbf{x}: d_i(\mathbf{x})=d} b(\mathbf{x}) \exp(\sum_{j \neq i} \phi_j d_j(\mathbf{x})) \exp(\phi_i d)) \\
&= -\ln(\sum_d y_d t_i^d),
\end{aligned}$$

where $y_d = \sum_{\mathbf{x}: d_i(\mathbf{x})=d} b(\mathbf{x}) \exp(\sum_{j \neq i} \phi_j d_j(\mathbf{x}))$ and $t_i = \exp(\phi_i)$. It takes some rewriting to find a familiar formula.

$$\begin{aligned}
-\ln(\sum_d y_d t_i^d) &= -\ln(\prod_d (t_i + z_d)) \\
&= -\sum_d \ln(t_i + z_d) \\
&= -\sum_d \ln(\exp(\phi_i) + z_d). \qquad (11)
\end{aligned}$$

These terms are concave as functions of ϕ_i and convex as functions of $\exp(\phi_i)$. Therefore, we have two options:

1. Take the first-order Taylor expansion to estimate $\exp(\phi_i)$ as in the implicit equations algorithm.
2. Use quadratic minorization to estimate ϕ_i. Namely, $-\ln(\exp(\phi_i) + z_d)$ is exact $k(x)$ (10) with $x = -\phi_i$, $\gamma_0 = z_d$ and $\gamma_1 = 1$.

Some of the steps could give problems, like finding the values for z_s. The Rasch model, however, proves that in practice these problems can be overcome. Another simple example is provided by a test with n items of the same difficulty. If the item responses are independent, the number of correct answers of a person, $\sum_i X_{pi}$, is a binomial random variable with probability

$$P(\sum_i X_{pi} = x|\delta) = \frac{\binom{n}{x}\exp(x\delta)}{\sum_{j=1}^{n}\binom{n}{j}\exp(j\delta)} \text{ for } x = 0, \ldots, n.$$

In this case, the denominator is the scaling constant $a(\delta)$. It follows from the Binomial Theorem that

$$a(\delta) = \sum_{j=1}^{n}\binom{n}{j} j\exp(\delta) = [1 + \exp(\delta)]^n,$$

where $\binom{n}{j}$ is indeed the elementary symmetric function of order j with unit arguments. It can be seen that $-\ln(a(\delta))$ has the same form as (11).

5. Interval algorithms

Minorization algorithms update the current likelihood value at every iteration by finding a better point. For concave functions, one can also choose to update the interval in which the optimum has to lie. The interval updating continues until this interval is very small. Multiple steps have to be taken before continuing with a new parameter, whereas in iterative minorization one can continue with another parameter whenever one chooses to.

The idea is that the slope indicates at which side of the optimum a point lies. If one starts with two points on both sides of the optimum, one has to estimate a new

point somewhere in the middle. This point then replaces the one of which the slope has the same sign. We discuss two methods to determine this new points. In Section 5.1 we discuss the False Position method, also known under the name *regula falsi*, which only uses the first-order derivatives to shrink the interval, whereas in Section 5.2 we discuss the Tent method, which also makes use of the value of the likelihood function. Both methods are one-dimensional.

5.1. False Position method

We start with two points at both sides of the optimum, of which we calculate the first-order derivatives. At every iteration we draw a straight line through the points $(\widehat{x_l}; \partial f(\widehat{x_l})/\partial x)$ and $(\widehat{x_r}; \partial f(\widehat{x_r})/\partial x)$. The new point x_s is the point where this line meets the horizontal axis. The sign of its slope indicates on which side of the optimum this point lies, and thus which point has to be replaced. If the slope is positive, this point replaces the point on the left side of the optimum. If the slope is negative, this point replaces the point on the right side of the optimum. Now, again a new point can be computed. In this way, iteratively the (one-dimensional) interval is shrank until it is small enough. Press, Flannery, Teukolsky, and Vetterling (1989) give the formula that is used to find x_s:

$$x_s = \widehat{x_l} - \frac{f'(\widehat{x_l})(\widehat{x_r} - \widehat{x_l})}{f'(\widehat{x_r}) - f'(\widehat{x_l})}.$$

For the Rasch model, the first-order derivative with respect to δ_j is

$$\frac{\partial l_{\mathbf{m}|\mathbf{n}}(\boldsymbol{\delta})}{\partial \delta_j} = -m_i + \sum_p \frac{\exp(-\delta_j)\gamma^{(j)}_{n_p-1}}{\gamma^{(j)}_{n_p} + \exp(-\delta_j)\gamma^{(j)}_{n_p-1}}. \tag{12}$$

Because all parameters are constant except for δ_j, it proves more convenient to write $l_{\mathbf{m}|\mathbf{n}}(\delta_j)$ instead of $l_{\mathbf{m}|\mathbf{n}}(\boldsymbol{\delta})$. The point $\delta_{j,s}$ can be found with:

$$\delta_{j,s} = \widehat{\delta_{j,l}} - \frac{\partial l_{\mathbf{m}|\mathbf{n}}(\widehat{\delta_{j,l}})}{\partial \delta_j}(\widehat{\delta_{j,r}} - \widehat{\delta_{j,l}}) \left(\frac{\partial l_{\mathbf{m}|\mathbf{n}}(\widehat{\delta_{j,r}})}{\partial \delta_j} - \frac{\partial l_{\mathbf{m}|\mathbf{n}}(\widehat{\delta_{j,l}})}{\partial \delta_j} \right)^{-1}.$$

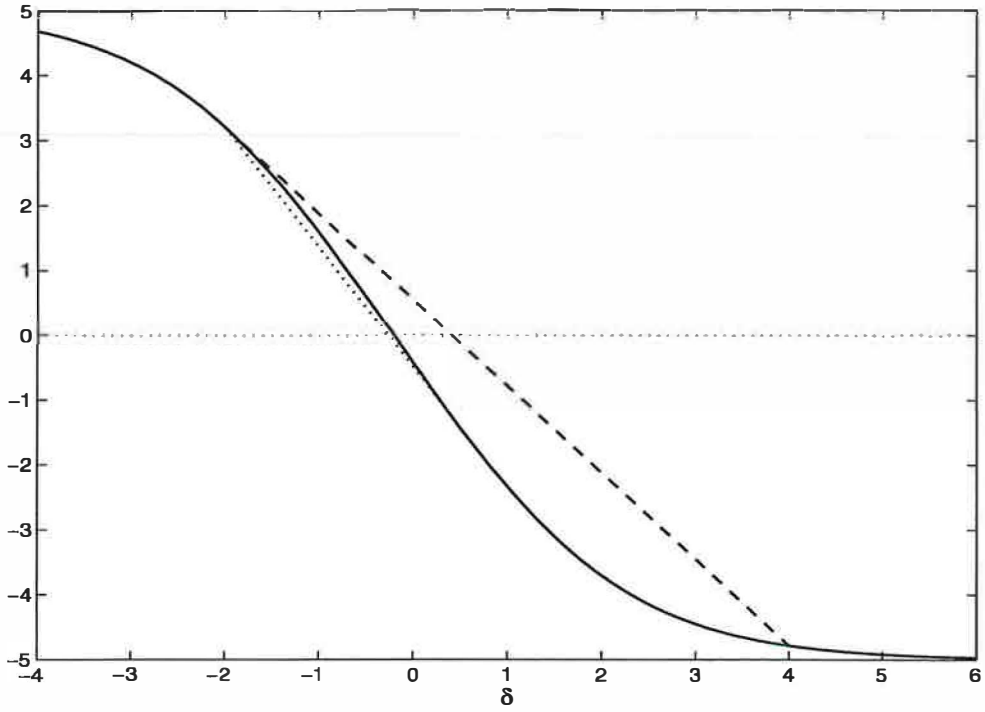

FIGURE 9.

Two steps of the False Position method with starting points -4 and 2.

Figure 9 gives an illustration of the False Position method. We start at the points with δ's equal to -2 and 4. We draw a straight line between these points. The new point, where the line meets the x-axis, is circa 0.4. This point replaces the right point with *delta* equal to 4, because the values of the slope of these points have the same sign. Now, we repeat this procedure by drawing a straight line between the points -2 and 0.4. This continues until a point is found which is close enough to the optimum. In other words, a point for which the value of the slope is small enough.

5.2. Tent method

In section 2.1, we saw that a convex function can be minorized by a tangent line (2). By taking two tangents at supporting points on opposite sides of the optimum,

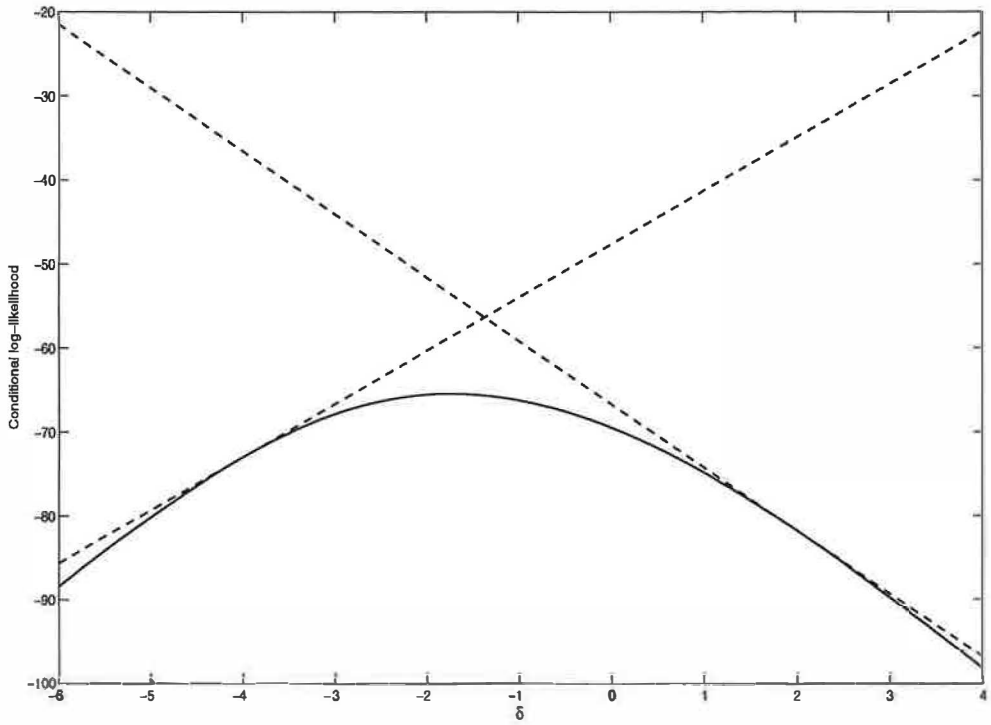

FIGURE 10.

Illustration of the Tent method with tangent points -4 and 2.

we can *cover* a concave function. These two tangents form a *tent* above the function. In literature, this tent is used to approximate a concave function. This procedure is known as the maximum error rule of the Sandwich algorithm (Rote, 1992). The supporting points of these two tangents lie on both sides of the optimum. The point where the two tangents meet, is the new point x_s. We compute the first-order derivative to see which point it replaces. To the best of our knowledge, it has not been used as an algorithm to find the optimum before. The point x_s can be found as follows:

$$x_s = f(\widehat{x_r}) - f(\widehat{x_l}) + \widehat{x_l}\frac{\partial f(\widehat{x_l})}{\partial x} - \widehat{x_r}\frac{\partial f(\widehat{x_r})}{\partial x}\left(\frac{\partial f(\widehat{x_l})}{\partial x} - \frac{\partial f(\widehat{x_r})}{\partial x}\right)^{-1}.$$

Figure 10 shows how such a tent looks like for the CLLF. With the left point

$\widehat{\delta_{j,l}}$, and the right point $\widehat{\delta_{j,r}}$, we construct two tangents of the following form:

$$T_{\mathbf{m|n}} = l_{\mathbf{m|n}}(\widehat{\delta_j}) + (\delta_j - \widehat{\delta_j})\frac{\partial l_{\mathbf{m|n}}(\widehat{\delta_j})}{\partial \delta_j}.$$

The point $\delta_{j,s}$ where the two tangents meet, is now computed as follows:

$$\begin{aligned}\delta_{j,s} = \quad & l_{\mathbf{m|n}}(\widehat{\delta_{j,r}}) - l_{\mathbf{m|n}}(\widehat{\delta_{j,l}}) + \widehat{\delta_{j,l}}\frac{\partial l_{\mathbf{m|n}}(\widehat{\delta_{j,l}})}{\partial \delta_j} \\ & - \widehat{\delta_{j,r}}\frac{\partial l_{\mathbf{m|n}}(\widehat{\delta_{j,r}})}{\partial \delta_j}\left(\frac{\partial l_{\mathbf{m|n}}(\widehat{\delta_{j,l}})}{\partial \delta_j} - \frac{\partial l_{\mathbf{m|n}}(\widehat{\delta_{j,r}})}{\partial \delta_j}\right)^{-1}.\end{aligned}$$

6. Newton-Raphson

Newton-Raphson is a well-known algorithm that is generally used to find the optimum of a function. It estimates this function with its second-order Taylor expansion at the supporting point, which is then optimized by derivation. Therefore, it works well when a function is close to quadratic. When a function is close to linear at the supporting point, it is more difficult to get a good estimation of the optimum. The reason is that the second-order derivative is close to zero; the optimum found by Newton-Raphson can have a larger distance to the real optimum than the old point. Therefore, there is no guarantee of convergence.

In Exponential Family Models, the (log)likelihood functions sometimes are quite linear in the tails, but quadratic when close to the maximum. Therefore in practice, one usually employs Newton-Raphson only after having taken some steps with a ascent algorithm which is guaranteed to converge.

We distinguish Newton-Raphson on the likelihood function with all parameters free (multidimensional), and the likelihood function for a single free parameter (one-dimensional). We briefly introduce both algorithms in the following sections.

6.1. *One-dimensional Newton-Raphson*

Newton-Raphson estimates a function by its second-order Taylor expansion from which the optimum, and thus new value, can be found:

$$g(x) = f(\widehat{x}) + f'(\widehat{x})(x - \widehat{x}) + 0.5f''(\widehat{x})(x - \widehat{x})^2.$$

The optimum of $g(x)$ is found by setting the first-order derivative equal to zero, and solving it for x:

$$x = \widehat{x} - \frac{f'(\widehat{x})}{f''(\widehat{x})}.$$

We already know the first-order derivative of the CLLF, see (12). The second-order derivative can be computed by

$$\frac{\partial^2 l_{\mathbf{m}|\mathbf{n}}}{\partial \delta_j \partial \delta_j} = -\sum_p \frac{\gamma_{n_p}^{(j)} \gamma_{n_p-1}^{(j)} \exp(-\delta_j)}{(\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)})^2}. \tag{13}$$

Figure 11 graphically illustrates the second-order Taylor expansion of the CLLF.

6.2. *Multidimensional Newton-Raphson*

If we choose to update all parameters at once, we use the following algorithm:

$$\boldsymbol{\delta}_n = \boldsymbol{\delta}_{n-1} - \mathbf{H}^{-1}\mathbf{g},$$

where $\mathbf{g}$ is the vector of first-order derivatives (Gradient), and $\mathbf{H}$ is the matrix of second-order derivatives (Hessian). The diagonal elements of the Hessian can be derived by (13), and the off-diagonal elements (i,j), for $i \neq j$, are equal to

$$\frac{\partial^2 l_{x|n}}{\partial \delta_i \partial \delta_j} = \sum_p \left(\frac{\gamma_{n_p-1}^{(i)} \gamma_{n_p-1}^{(j)} \exp(-\delta_i - \delta_j)}{(\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)})^2} - \frac{\gamma_{n_p-2}^{(ij)} \exp(-\delta_i - \delta_j)}{\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)}} \right).$$

With this method, a whole matrix of second-order derivatives has to be calculated and inverted, which takes quite some time. Computation of this Hessian lets the convergence time of Newton-Raphson be quadratic in the number of items. Inverting

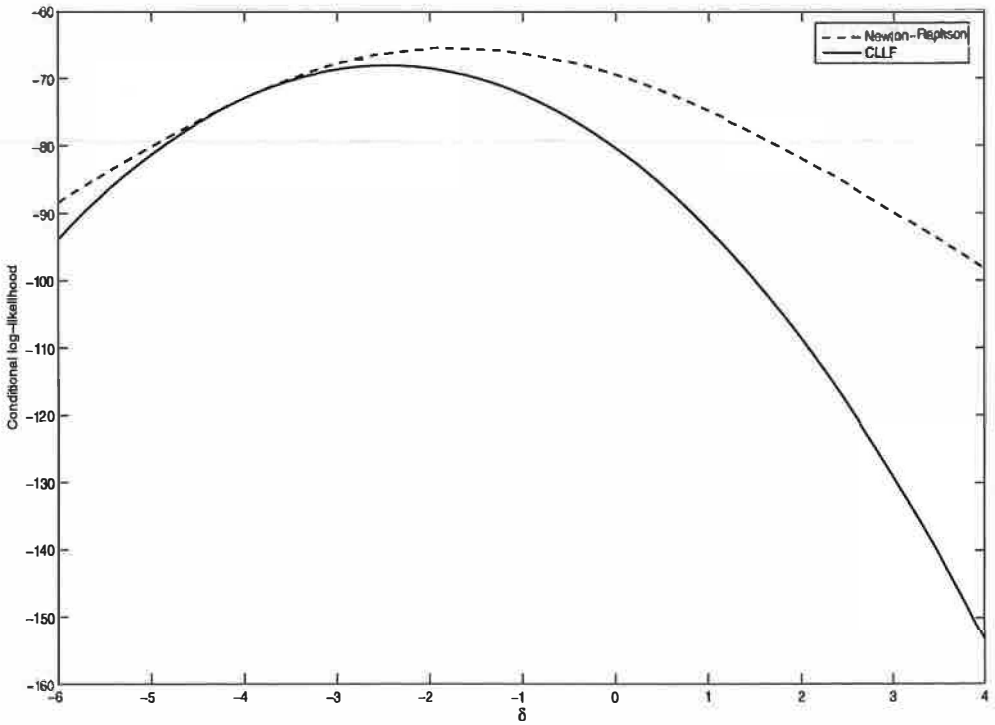

FIGURE 11.

CLLF estimated by its second-order Taylor expansion.

the Hessian can be avoided, by solving the system of linear equations

$$\mathbf{H}(\boldsymbol{\delta}_{n-1} - \boldsymbol{\delta}_n) = \mathbf{g}.$$

The total count of solving this system even is of order N^3 (Press et al., 1989).

7. Simulation study

To give an idea how fast every algorithm works, we perform a small simulation study. We simulate data to test the algorithms for the Rasch model. Before the simulation, we first show that the performance of the minorizations depends on the (one-dimensional) shape of the graphs. Then, we study what determines these shapes, so we know what we have to vary to give a complete comparison between all algorithms. After the simulations, we test the methods on three empirical datasets.

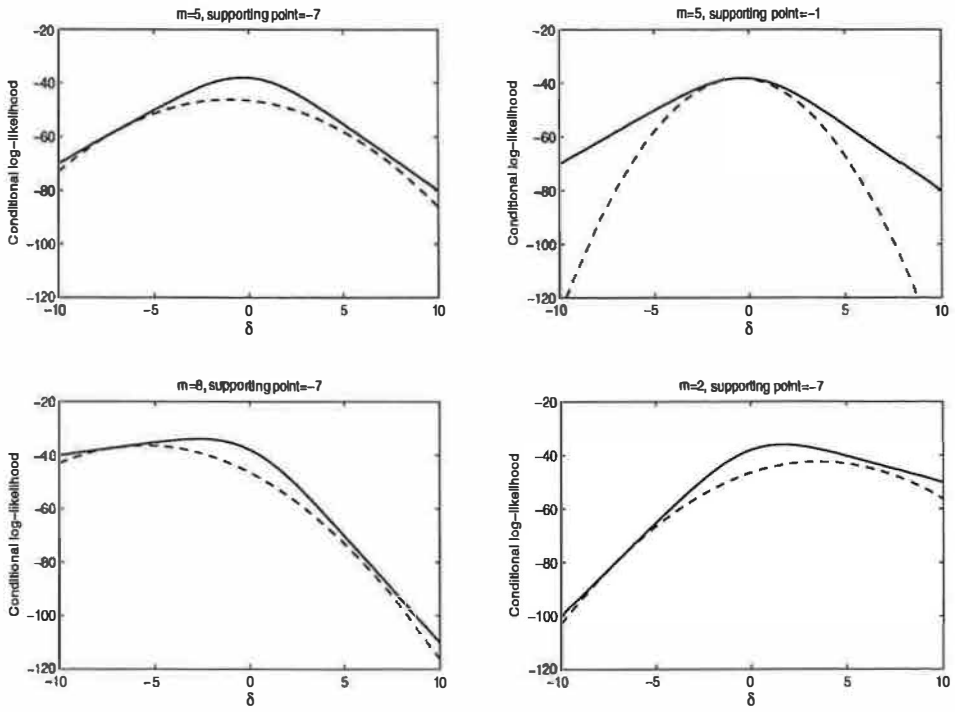

FIGURE 12.

Difference in steps of quadratic minorization due to other curvatures.

Figures 12 and 13 show that the fit and the optima of the two minorizations are influenced by the shape of the graphs. The quadratic minorization seems to have an optimum closer to the real optimum if the graphs are more symmetric, whereas in the implicit equations algorithm, the steepness of the graph plays an important role. In two ways, we determine the factors that influence these shapes. First, we derive these from the CLLF; secondly, we make some plots, varying all kinds of parameters. We do not take the ability parameters into account, because a change in mean or in variance of the population can also be created by changing the difficulty parameters.

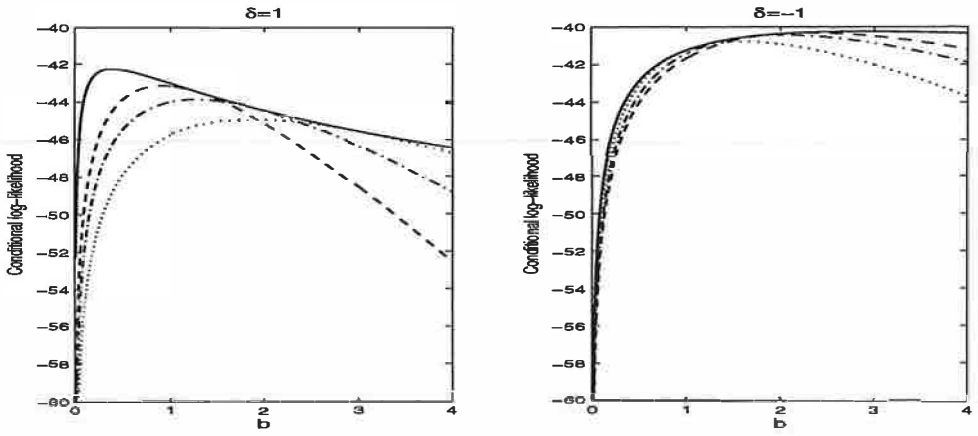

FIGURE 13.

Difference in steps of implicit equations due to other curvatures.

Recall the CLLF (6)

$$\begin{aligned} l_{\mathbf{m}|\mathbf{n}}(\boldsymbol{\delta}) &= -\sum_{i} m_i\delta_i - \sum_{p} \ln(\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)}) \\ &= -\sum_{i\neq j} m_i\delta_i - m_j\delta_j \\ &\quad -\sum_{p}\left[\ln\left(1+\exp\left(\ln\left(\frac{\gamma_{n_p-1}^{(j)}}{\gamma_{n_p}^{(j)}}\right)-\delta_j\right)\right) - \ln\left(\gamma_{n_p}^{(j)}\right)\right], \end{aligned} \tag{14}$$

and its first-order derivative (12)

$$\frac{\partial l_{\mathbf{m}|\mathbf{n}}}{\partial \delta_j} = -m_j + \sum_{p} \frac{\gamma_{n_p-1}^{(j)}\exp(-\delta_j)}{\gamma_{n_p}^{(j)} + \exp(-\delta_j)\gamma_{n_p-1}^{(j)}},$$

of which the limits are

$$\lim_{\delta_j\to-\infty} \frac{\partial l_{\mathbf{m}|\mathbf{n}}}{\partial \delta_j} = P - m_j = P(1-\frac{m_j}{P}), \tag{15}$$

$$\lim_{\delta_j\to\infty} \frac{\partial l_{\mathbf{m}|\mathbf{n}}}{\partial \delta_j} = -m_j = -P\left(\frac{m_j}{P}\right), \tag{16}$$

where P is the number of persons. To focus more on the proportion of persons, the

CLLF can also be written as

$$
\begin{aligned}
&-\sum_i m_i\delta_i - \sum_s r_s \ln(\gamma_s^{(j)} + \exp(-\delta_j)\gamma_{s-1}^{(j)}) \\
&= P\left(-\sum_i \frac{m_i}{P}\delta_i - \sum_s \frac{r_s}{P} \ln(\gamma_s^{(j)} + \exp(-\delta_j)\gamma_{s-1}^{(j)})\right),
\end{aligned}
$$

where s are the possible scores, and r_s the number of persons who have score s. m_i/P represents the proportion of persons who made item i correctly, whereas r_s/P represents the proportion of persons who have score s. One can see that the limits of the derivative are influenced by the former.

Using these functions, we can describe three factors that seem important for the shape of a one-dimensional graph (for δ_j free):

1. The number of persons; more persons cause the graph to be lower, and steeper. This can be seen from the limits of the derivative (15) and (16). While P increases, the ratio m_j/P remains the same.
2. The difficulty parameter δ_j; a smaller δ_j causes a higher proportion of persons who have made the item correctly. The larger m_i/P, the more the graph leans to the right, which can be seen from (15) and (16).
3. The other difficulty parameters; a larger δ_i $(i \neq j)$ causes the graph to move down because of the first term, and to move upward, because all γ-functions, which are monotonic increasing in $\exp(-\delta_i)$, decrease. The sum of these two changes determines the vertical shift. It also causes the graph to shift horizontally, and to change shape. The direction of these changes, however, cannot be predicted without more knowledge. The direction for instance depends on the number of people who have a certain score, and the size of the other δ_i's.

This score-distribution depends on the items in relation to the population, which includes the three previous factors. The γ-functions are single peaked, see Corollary

5. If more persons score either very good or very bad, r_s decreases for average s, but increases for low and high s. The fact that the γ-functions for these values are generally lower, implies that the graph becomes lower. However, whereas the change of the first-order derivative depends on the relation between γ_{n_p} and γ_{n_p-1}, its limits are not affected.

More items do not change the shape of the graph much, because they do not affect the limits of the derivative. However, the terms $\ln(\gamma^{(j)}_{n_p-1}/\gamma^{(j)}_{n_p}) - \delta_j$ (14) change for every p, which means that the shape around the optimum is changed in a non-predictable way. The graph becomes lower for two reasons: the γ-functions increase, and per extra item k the graph moves down by $m_k\delta_k$.

To get more certainty, we make six one-dimensional plots for δ_j. We vary the number of persons, the number of items, m_j, and the δ_i's ($i \neq j$) to get a better insight. In every plot, we cover the graphs with straight lines which represent the limits of the derivative.

The picture of the graph in the upper left corner of Figure 14 is the basic picture. Here, we have 10 persons, 10 items, $m_j = 5$ and all δ_i equal to zero. For all other graphs, we change one aspect of the function. We discuss the effect of each change separately, except for the changes in δ's. Namely, we compare the changes caused by diversification and increasing under one bullet.

- For the graph in the upper middle the number of persons has been doubled. The shape sustains that more persons let the graph be steeper and lower.
- In the picture on its right, the number of items has been doubled, which results in a lower graph, which is shifted to the right. As expected, the limits of the derivatives are not influenced.
- In the picture in the down left corner, m_j is equal to 9, which clearly causes the

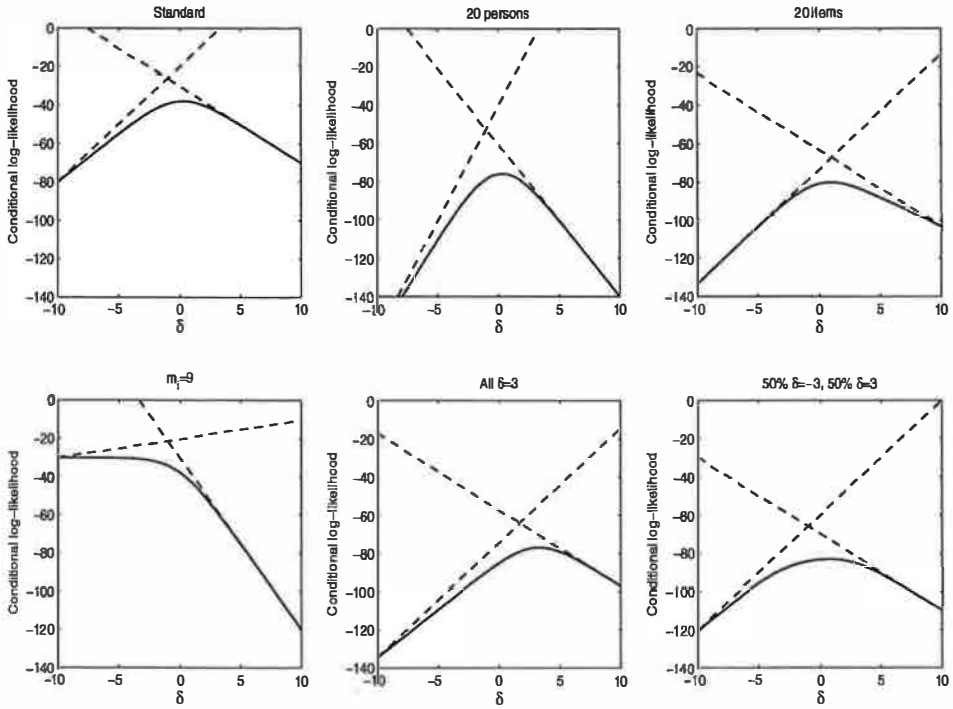

FIGURE 14.

Shape of the one-dimensional graph with changing parameters.

asymptotes to lean more to the right.

- If we compare the last two pictures, we see that diversifying the δ's (down right) causes the graph to be lower and flatter than increasing all δ's (down middle). On the other hand, the latter causes the graph to move to the right, whereas the former does not cause a significant horizontal shift.

In this simulation study, we perform every simulation one hundred times on a Toshiba Notebook with a 1.60 GHz Intel Pentium M processor. The study is divided in five parts. The root simulation is:

- 1000 persons,
- 100 items,

- All $\delta = 0$.

The ability parameters $\boldsymbol{\theta}$ of the persons are drawn from a normal distribution with mean zero and variance 1. We create a matrix of answers which are either right or wrong, where we use the probability given in (1). The starting values for all algorithms are zero for all δ's.

In the first simulation, we vary the number of persons to see if a steeper function increases the speed of convergence:

- 100 persons;
- 10000 persons.

In the second simulation, we vary the number of items:

- 20 items;
- 60 items.

In the third simulation, we investigate the influence of the m's on the speed. To isolate the effect of the m's, we want the estimated δ's to be the same as before (around zero), and the m's to be larger. Taking all δ equal to 1 provides the solution. Because they are all estimated relatively to the first item, and δ_1 is set equal to zero in the estimation, the only difference are the m's. To gain insight in what happens to the speed if the δ's get larger or smaller, in the fourth simulation, we keep δ_1 equal to zero, but vary the other δ's:

- δ_1=0, δ=-1;
- δ_1=0, δ=1.

The fifth simulation investigates the influence of diversification in the δ's by again

taking δ_1 equal to 0, 50 δ's equal to -1, and 49 equal to 1.

We test the six one-dimensional algorithms, including implicit equations accelerated by Aitken's method, together with multidimensional Newton-Raphson. They all stop when the progress in the log-likelihood is smaller than 10^{-5}, and the absolute change per b_i is smaller then 10^{-4}. The evaluation is performed after all parameters have been updated once.

At every run, we have to recalculate the γ-functions a large number of times. To save time, we recursively compute them out of the old γ-functions, which we rebuild after updating all items. We shortly describe how all algorithms are implemented in Visual Basic:

Minorization algorithms

In every iteration one difficulty parameter is updated. After this update, we continue with the next parameter.

Implicit equations accelerated by Aitken's method

For notational convenience, we refer to this method by Aitken. Two new values are computed for one parameter. A third value is only computed if the second step is smaller than the first step. In this case, we calculate the new likelihood. If this likelihood is larger than the old likelihood, the Aitken-value is the new value. If the second step is smaller than the first step or if the likelihood is smaller, we take the second value as the new parameter value. After this update, we continue with the next parameter.

Interval algorithms

For every parameter update, the difficulty lies in finding a value on the other side of the optimum. We start at the old value and perform a line search. In this search, we take steps in the direction of the optimum. If the new value, is on the same side, it replaces the old value. If it is on the other side, the algorithm starts. The steps are 0.02 for the Tent method and 0.15 for the False Position method, because these have performed well in pre-testing. Another idea which we have not used is to vary these steps. For instance, it might be a good idea to take smaller steps when being closer to the optimum. The parameter is updated when the absolute value of the first-order derivative of a new point is smaller than 10^{-3}.

One-dimensional Newton-Raphson

Like in the minorization algorithms, in every iteration one difficulty parameter is updated. If the new likelihood value (computed after updating all parameters) is smaller than the old value, the algorithm stops.

Multidimensional Newton-Raphson

Multidimensional Newton-Raphson involves solving a system of linear equations. We solve them by computing the Choleski-decomposition of the Hessian, and then using back-substitution. More details can be found in Press et al. (1989). All values are updated in one iteration. If the new likelihood value is smaller than the old value, the algorithm stops.

In every run, we measure the CPU-time an algorithm lasts, and how many iterations it needs. We also keep track of the algorithm which is fastest in time and number of iterations. We do not take into account multidimensional Newton-

Raphson, because the starting values have been chosen in the advantage of this algorithm. Namely, they are in the approximately quadratic area of the likelihood, which makes this algorithm always the fastest, both in speed as well as in number of iterations. The number of iterations counts here the number of times all parameters are updated. In case of updating the algorithm which uses the least number of iterations, there is a small bias. Namely, if the number of iterations is smaller than the number of iterations of a previous algorithm, we replace the one which is fastest. It follows that when the number of iterations is equal, the algorithm which has been run first, remains the fastest. Therefore, the earlier an algorithm is run, the more likely it is that it comes out as the one with the least number of iterations.

7.1. Results

The results are presented in five separate tables, each corresponding to a simulation. We first discuss only the guaranteed converging algorithms. Then, we compare them shortly with Newton-Raphson, and determine when the Aitken Accelator helps to fasten the implicit equations algorithm. After the simulations, we test the algorithms on three empirical datasets. In the tables, the most striking results are oblique.

In Table 1, the False Position method is the fastest algorithm, whereas the Tent method is the slowest algorithm. We see that the average number of iterations of the quadratic minorization is about the same as the average number of iterations of the interval algorithms. This finding suggests that the optimum of the minorization is quite close to the real optimum.

More persons cause the (one-dimensional) graphs to be steeper. The speed of the interval algorithms decreases, because it takes longer to come close to the tops of these graphs per iteration. Namely, in the implementation we have chosen the slope

TABLE 1.

Speed and number of iterations when varying the number of persons.

# persons	100	1000	10000
Average time			
Impl eq	1.777	1.856	1.794
Quadratic	4.990	4.402	4.101
False Pos	1.389	*1.396*	*1.433*
Tent	6.281	*7.533*	*9.255*
Average # iterations			
Impl eq	420.06	422.73	398.32
Quadratic	188.69	169.24	157.60
False Pos	184.45	167.18	155.74
Tent	217.12	167.28	155.58

as criterium of closeness to continue with the next parameter. However, this choice does bring down the number of iterations, because the values after an iteration are more accurate. The algorithm which profits most from the steeper graphs is the quadratic minorization. The speed of the implicit equations algorithm does not change significantly. For 10000 persons, however, the number of iterations becomes fairly smaller.

A remarkable result is that with hundred persons the number of iterations of the Tent method is significantly larger than the number of iterations of the False Position method, while they are both programmed to come close to the top per iteration. We can think of two possible causes:

1. The smaller steps of the line search of the Tent method let this method stop at values further from the optimum than the False Position method.
2. The criterium for closeness is not strong enough when the number of persons is

ten times smaller. The decrease in the number of persons lets the slopes of the derivative also become ten times as small, see (15) and (16).

We test both assumptions by two new simulations with the interval algorithms. In the first simulation, we swap the steps of the line search, and in the second, we let the criterium of the slope be ten times smaller. We keep track of the one with the least number of iterations. If both numbers are the same, we add one for both algorithms.

TABLE 2.

Speed and number of iterations with swapped line search and more severe criteria.

	Line search	Criteria
Average time		
False Pos	1.377	1.452
Tent	7.780	7.557
Average # of iterations		
False Pos	184.44	185.11
Tent	217.17	185.24
# least iterations		
False Pos	91	60
Tent	11	59

Table 2 shows that the line search has no significant influence on the number of iterations. This number remains fairly larger for the Tent method, and the False Position method needs the same number or less in 91 runs. The more severe criterium for the slope, however, lets both algorithms use around the same number of iterations. This criterium seems more important for the Tent method. A possible explanation is that in every iteration, the False Position jumps to the maximum when being close to it, whereas the Tent method takes smaller steps, and is inclined to stop further from the one-dimensional optimum.

TABLE 3.

Speed and number of iterations when varying the number of items.

# items	20	60	100
Average time			
Impl eq	0.027	0.457	1.856
Quadratic	0.044	1.000	4.402
False Pos	0.023	0.359	1.396
Tent	0.118	1.947	7.533
Average # iterations			
Impl eq	110.40	269.41	422.73
Quadratic	42.70	106.90	169.24
False Pos	40.30	104.79	167.18
Tent	40.31	105.37	167.28

Table 3 shows that the influence of the number of items on the speed is more than linear. The interval algorithms seem to be most stable for these changes. For instance, if we divide the speed at 100 items by the speed at 20 items, the outcome is for the Tent method only 63.81, whereas for the quadratic minorization, the outcome is 100.05.

As previously stated, the simulations with all δ equal to one isolate the effect of smaller m's. They become lower, so the graphs lean to the left. In Table 4, it can be seen that all algorithms become faster, except for the quadratic minorization, which needs a little more iterations. Figure 12 suggests that the speed of this algorithm is less affected by the distance to the optimum (top graphs), as it is by the skewness (lower graphs), which heavily influences the distance to the optimum of the new point. Here, this distance remains the same. The skewness, however, is larger, because the differences between the absolute values of the limits of the first-order derivatives ($|(P-m_i)-(m_i)|$) become larger. Namely, this difference is close to zero if the mean of the θ's is about the same as the value of the δ's, which causes every

TABLE 4.

Speed and number of iterations when varying the m's.

δ's	0	1
Average time		
Impl eq	1.856	*1.054*
Quadratic	4.402	4.474
False Pos	1.396	1.185
Tent	7.533	6.083
Average # iterations		
Impl eq	422.73	242.31
Quadratic	169.24	171.98
False Pos	167.18	141.53
Tent	167.28	141.31

item to be made correctly by circa 50% of the persons. In contrary to earlier intuition, this skewness only slightly slows the algorithm down. In the next simulation, we show that the distance to the optimum plays a more significant role than the skewness.

The implicit equations algorithm now is the fastest. Although it still uses most iterations, this number is so much reduced that it is now more than compensated by the low time per iteration. This remarkable result can be explained by Figure 13. This figure shows that the graph is quite different for positive and negative δ's. For higher δ's, the graph is much steeper in the b-parameterization, which let the optimum of the implicit equations algorithm be closer to the real optimum.

For all algorithms, the speed and number of iterations become larger when $\delta_1 = 0$ and the other δ are not equal to zero. This increase can be explained by the fact that the starting values are now further away from the optimum.

In contrast to the previous results, the number of iterations of quadratic minorization is significantly higher than the number of iterations of the interval al-

TABLE 5.

Speed and number of iterations when varying the optima.

δ ($\delta_1 = 0$)	0	1	-1
Average time			
Impl eq	1.856	2.120	*5.750*
Quadratic	4.402	*8.849*	*9.281*
False Pos	1.396	2.353	2.352
Tent	7.533	11.465	11.512
Average # iterations			
Impl eq	422.73	491.89	*1333.58*
Quadratic	169.24	*335.29*	*334.99*
False Pos	167.18	272.05	271.83
Tent	167.28	271.86	272.28

gorithms. Starting at a greater distance from the optimum apparently does let the optimum of the minorization be further away from the real optimum. Our earlier intuition that the skewness plays a more important role than the distance between the optimum and the supporting point seems to be wrong.

In contrast to the other algorithms, the results for the implicit equations algorithm differ a lot for positive and negative δ. If we let δ_1 be equal to zero, and we set all other δ equal to one, the implicit equations algorithm becomes only a little slower, whereas all other algorithms last about twice as long. This result can be explained since the distance from the starting values towards the optimum on the b-scale is not so big, and also the graph is steeper, as previously stated. If we instead let all other δ be equal to minus one, the implicit equations algorithm becomes much slower, because the distance between the starting values and the optimum on the b-scale becomes much larger. Figure 13 shows that this distance is larger, and that the shape of the graph is also less steep.

For all algorithms the convergence time is lower than the average time of the

TABLE 6.

Speed and number of iterations when diversifying the δ's.

δ ($\delta_1 = 0$)	0	50 $\delta = -1$ 49 $\delta = 1$
Average time		
Impl eq	1.856	2.854
Quadratic	4.402	7.865
False Pos	1.396	2.135
Tent	7.533	10.342
Average # iterations		
Impl eq	422.73	647.66
Quadratic	169.24	293.85
False Pos	167.18	241.98
Tent	167.28	242.17

two previous simulations. The diversity of the δ's lets the values of the parameters converge faster to their optima.

Empirical datasets

The next simulations are performed with empirical datasets. The first set stems from a larger dataset (Baumert, Heyn, Köller, & Lehrke, 1992). Rost (1996) has selected this dataset for expository purpose. The other two are the results from state examinations for Dutch as a second language.

The time measurements in Table 7 are less secure than before, because each observation stems from only one run. However, because the results are quite similar to what we previously have observed, they provide a strong indication that the False Position method indeed is the fastest. In contrary to the speed, the number of iterations still is reliable. Again, both interval algorithms use the least number of iterations. The number of iterations of the quadratic minorization is higher for a fac-

TABLE 7.

Results from real datasets.

# items	15	40	12
# persons	300	2197	3262
Time			
Impl eq	0.111	0.591	0.070
Quadratic	0.090	0.722	0.020
False Pos	0.020	0.220	0.010
Tent	0.100	1.362	0.060
# iterations			
Impl eq	360	635	347
Quadratic	118	155	65
False Pos	64	107	40
Tent	65	105	40

tor between 1.5 and 2. The implicit equations algorithm uses by far most iterations, but its time per iteration is again very small.

Not guaranteed converging algorithms

TABLE 8.

Selection of results for Aitken and Newton-Raphson.

δ	$\delta = 0$	$\delta = 1$	$\delta_1 = 0$ $\delta = -1$	$\delta_1 = 0$ $\delta = 1$	$\delta_1 = 0$ δ_a [1]
Avg. time					
Aitken	1.46744	1.22429	2.26188	2.2666	2.04614
1-dim N-R	1.81953	1.55879	2.87431	2.8763	2.54995
M-dim N-R	0.86892	0.8415	0.72986	0.74756	0.93748
Avg. # its.					
Aitken	167.18	141.48	271.82	272.05	242.49
1-dim N-R	167.2	143.36	272.26	272.42	240.73
M-dim N-R	3	3	3.22	3.3	4

Table 8 shows the results for the simulations with the three remaining algorithms in which the δ's have been varied. As expected, multidimensional Newton-Raphson is by far the fastest algorithm, both in speed as well as number of iterations. The accelerated implicit equations algorithm is the fastest of all one-dimensional algorithms. However, the Aitken Accelator does not fasten the implicit equations algorithm when the δ's are positive. Then, there is relatively less gained in the number of iterations, probably because the implicit equations algorithm already is very fast. The implicit equations algorithm accelerated by Aitken and one-dimensional Newton-Raphson need about the same number of iterations as the interval algorithms. This observation points out that all these algorithms come close to the one-dimensional optimum in every iteration. The interval algorithms are created in this way, but for Aitken and one-dimensional Newton-Raphson this fact is less obvious.

8. Conclusion

The goal of this thesis was to improve the current package OPLM. Here, a combination of implicit equations and multidimensional Newton-Raphson is used.

We have found that the implicit equations algorithm is in fact a minorization algorithm. This algorithm needs most iterations in general, whereas the time per iteration is the smallest. This algorithm is unstable in relation to the difficulty parameters. More difficult items let the algorithm become faster. If more items are made correctly, the Aitken Accelerator helps creating a much faster algorithm.

Multidimensional Newton-Raphson becomes problematic in case there are many items. The number of computations of the algorithm increases quadratically with the number of items, and the solution of a linear system becomes unstable for a large number of items. However, if the algorithm converges, it does so rapidly.

[1] δ_a: 49 $\delta = -1$, 50 $\delta = 1$

To combine the best properties of both algorithms, we have presented four alternative one-dimensional algorithms:

1. Quadratic minorization;
2. False Position;
3. One-dimensional Newton-Raphson;
4. Tent method,

of which the first three are quadratic approximations of the function.

We have shown that the quadratic minorization we used minorizes the *difficult* terms the best of all possible quadratic minorizations. The quadratic minorization algorithm works well if the starting values are close to the optimum. Then, it is about as fast as the interval algorithms in number of iterations. If the starting values are further away, the distance between the optimum of the minorization and the one-dimensional optimum becomes fairly large.

The False Position method proves to be the fastest algorithm for the Rasch model. This algorithm needs about the same number of iterations as the other interval algorithm, the Tent method. This result is not surprising since they are both programmed to continue updating until they are very close to the one-dimensional optimum. Per iteration, however, the False Position method is circa a factor five faster than the Tent method. Here, the gain of knowing the value of the CLLF does not counterbalance the cost of calculation time. The Tent method is the slowest monotonic converging algorithm. For all algorithms, the speed is about the same for every simulation, not including a different number of items, except for this method. Criteria for being close to the optimum which depend on the number of persons keep the speed more stable.

The simulation shows that the speed of the algorithms, and the number of iter-

ations greatly depend on the shape of the function. We have shown this dependence for the minorization algorithms, and the observations make clear that this dependence also holds for the other algorithms.

Final remarks

We now give some ideas to improve the convergence rate of the other algorithms. We also provide some suggestions for further research.

If a large proportion of the items is made incorrectly, the implicit equations algorithm becomes faster. This algorithm converges faster for difficult items. It might be interesting to see what happens if, instead of the first item parameter, one sets the difficulty parameter of the item which is made correctly the most equal to zero. The other items are estimated relatively to this item, and therefore their estimates are larger in the δ-scale, and smaller in the b-scale. In this method, one is sure that the differences between the starting values and the optima are not larger than one. A second option is to use another parameterization for easy items. For these items, instead of using $b_i = \exp(-\delta_i)$, one uses $b_i = \exp(\delta_i)$.

It might also be interesting to construct a multidimensional minorizing function to update all parameters at the same time. For the Joint Maximum Likelihood, an algorithm has been proposed by Groenen et al. (2003). However, for CML it is not clear how and if it can be done.

We have shown that the Aitken Accelerator in certain cases helps to fasten the implicit equations algorithm. Other algorithms may also be helped by this Accelerator. If for instance two values of the Tent method are known which lie on the same side of the optimum, this optimum can be approximated with Aitken's method.

Another suggestion to create a fast algorithm is to use a combination of multiple algorithms. For instance, the two interval algorithms now begin with a line search

starting at the current point, and then update the parameter until being close enough to the optimum. It might be interesting to see what happens if one starts with two points very far away on both sides of the optimum, and calculates one new value with one of the methods. We expect this value to lie in the quadratic area around the optimum. One can use this point together with the current point in the Secant method (Press et al., 1989), which is quite similar to the False Position method, but does not require that the two points are on the same side of the optimum. Another option is to use one-dimensional Newton-Raphson at this point. A second example follows from the problem that multidimensional Newton-Raphson is not guaranteed to converge. Therefore, it is wise to first take a few steps with a minorization or interval algorithm to get closer to the optimum, before using Newton-Raphson. In this case, it is interesting to know how many iterations of the monotonic algorithm are needed, and which algorithm takes these steps the fastest.

Some other ideas for further investigating the performance of these algorithms are:

- Use other accelerators;
- Compare the algorithms with other monotonically converging algorithms;
- Test the methods for other models.

References

Aitken, A. (1926). On Bernoulli's numerical solution of algebraic equations. *Proc. Roy. Soc. Edinburgh*, *46*, 289-305.

Andersen, E. B. (1980). *Discrete statistical models with social science applications.* Amsterdam: North-Holland Publishing Company.

Baumert, J., Heyn, S., Köller, O., & Lehrke, M. (1992). *Naturwissenschaftliche und psychosoziale Bildungsprozesse im Jugendalter (BIJU)-Testdokumantation*

[Scientific and psychosocial developmental processes in young children]. Kiel, Germany: IPN.

Borg, I., & Groenen, P. J. F. (1997). *Modern multidimensional scaling: Theory and applications.* New York: Springer.

De Leeuw, J. (1994). Block-relaxation algorithms in statistics. In W. L. H. H. Bock & M. M. Richter (Eds.), *Information systems and data analysis* (p. 308-325). Berlin: Springer-Verlag.

Fischer, G. H. (1974). *Einfuhrung in die Theorie psychologischer Tests.* Bern: Verlag Hans Huber. ((Introduction to the theory of psychological tests.))

Groenen, P. J. F., Giaquinto, P., & Kiers, H. A. L. (2003). *Weighted majorization algorithms for weighted least squares decomposition models* (Econometric Institute Report No. EI 2003-9). Erasmus University Rotterdam.

Heiser, W. J. (1995). Convergent computation by iterative majorization: Theory and applications in multidimensional data analysis. In W. J. Krzanowski (Ed.), *Recent advances in desriptive multivariate analysis* (p. 157-189). Oxford University Press.

Hunter, D., & Lange, K. (2004). A tutorial on MM algorithms. *American Statistician, 58, Issue 1,* 30-37.

Kiefer, J., & Wolfowitz, J. (1956). Consistency of the maximum likelihood estimator in the presence of infinitely many nuisance parameters. *Annals of Mathematical Statistics, 27,* 887-906.

Lange, K., Hunter, D., & Yang, I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics, 9,* 1-20.

Lehmann, E. L. (1986). *Testing statistical hypotheses* (Second ed.). New York: John Wiley & Sons.

Neyman, J., & Scott, E. L. (1948). Consistent estimates based on partially consistent observations. *Econometrica, 16*, 1-32.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. (1989). *Numerical recipes: The Art of Scientific Computing (Fortran Version).* Cambridge: University Press.

Rasch, G. (1960). *Probabilistic models for some intelligence and attainment tests.* Copenhagen: The Danish Institute of Educational Research. (Expanded edition, 1980. Chicago, The University of Chicago Press)

Rost, J. (1996). *Testtheory, testkonstruction [test theory, test construction].* Bern, Germany: Verlag Hans Huber.

Rote, G. (1992). The convergence rate of the sandwich algorithm for approximating convex functions. *Computing, 48*, 337-361.

Verhelst, N. D., & Verstralen, H. H. F. M. (1997). *Modeling sums of binary responses by the partial credit model* (Tech. Rep. No. 97-7). CITO: Arnhem.

9. Appendix

Proof of Theorem 1

In this proof, we consider polynomials of the form $ax^k + bx + c$ for $k \geq 2$, even and finite. We proof Theorem 1 and its corollaries.

Theorem 3. *For a given function, consider minorizing functions of the following form:*

$$m_{ik}(x, \hat{x}) = a_i x^k + b_i x + c_i,$$

where $\hat{x}$ is the supporting point and k is a fixed integer constant greater than or equal to 2, that is even and finite. There is no value $y \neq \hat{x}$ such that $m_{ik}(y, \hat{x}) = m_{jk}(y, \hat{x})$, for $i \neq j$. Furthermore, for any two such functions one minorizes the other.

Proof. Suppose target function $f(x)$ has two minorizing functions, $g(x, \hat{x})$ and $h(x, \hat{x})$, with the following properties:

$$g(x, \hat{x}) = ax^k + bx + c,$$

$$h(x, \hat{x}) = rx^k + sx + t,$$

$$g(\hat{x}, \hat{x}) = h(\hat{x}, \hat{x}) = f(\hat{x}),$$

$$g'(\hat{x}, \hat{x}) = h'(\hat{x}, \hat{x}) = f'(\hat{x}),$$

for $k \geq 2$, even and finite. We first rewrite $h(x, \hat{x})$ to make it easier to compare it with $g(x, \hat{x})$. Then, we show that the difference between the two functions is only zero at the supporting point, and that at all other points, one function minorizes the other.

Assume the difference between a and r is equal to z, thus $r = a + z$. Now we can write:

$$h(x, \hat{x}) = (a + z)x^k + sx + t,$$

$$h'(x, \hat{x}) = k(a + z)x^{k-1} + s.$$

Setting the first-order derivatives $g'(\hat{x}, \hat{x})$ and $h'(\hat{x}, \hat{x})$ equal, and solving for s gives:

$$k(a + z)\hat{x}^{k-1} + s = ka\hat{x}^{k-1} + b \Rightarrow s = b - kz\hat{x}^{k-1}.$$

t can now be found by setting $h(\hat{x}, \hat{x})$ equal to $g(\hat{x}, \hat{x})$:

$$(a + z)\hat{x}^k + (b - kz\hat{x}^{k-1})\hat{x} + t = a\hat{x}^k + b\hat{x} + c \Rightarrow t = c + z(k - 1)\hat{x}^k.$$

Now, we can derive at which points $g(x, \widehat{x})$ and $h(x, \widehat{x})$ meet, by solving

$$d(x, \widehat{x}) = h(x, \widehat{x}) - g(x, \widehat{x}) = zx^k - zk\widehat{x}^{k-1}x + z(k-1)\widehat{x}^k = 0.$$

By definition, a solution of this equation is $\widehat{x}$ ($z \neq 0$). Because the first-order derivative of $d(x, \widehat{x})$ is zero at $\widehat{x}$, and $d(x, \widehat{x})$ is monotonic, this solution is unique. In other words, $\widehat{x}$ is the only point where the two functions meet. Because both functions are continuous, $g(x, \widehat{x})$ minorizes $h(x, \widehat{x})$ for positive z, and $h(x, \widehat{x})$ minorizes $g(x, \widehat{x})$ for negative z. □

From the latter theorem and its proof follow two corollaries:

Corollary 3. *If the minorization $m_{ik}(x, \widehat{x})$ of a certain target function has a second supporting point, this second supporting point is unique, and this is the only function with multiple supporting points for given k.*

Corollary 4. *The unique minorizing function $m_{ik}(x, \widehat{x})$ with two supporting points is better than any other minorizing function of the same order k.*

Proof. Let the minorization of $f(x)$, $g(x, \widehat{x})$, have a second supporting point called $\overline{x_k}$. We want to know whether $h(x, \widehat{x})$, being different from $g(x, \widehat{x})$, can minorize $f(x)$, and touch $f(x)$ at two points, including $\widehat{x}$. We have to consider two cases:

1. $g(x, \widehat{x})$ minorizes $h(x, \widehat{x})$;
2. $h(x, \widehat{x})$ minorizes $g(x, \widehat{x})$.

We show that in the former case, $h(x, \widehat{x})$ does not minorize $f(x)$, and in the latter case, it cannot have a second supporting point. The fact that other minorizing functions of the same form cannot have a second supporting point, implies that this point is unique. Because the minorizing function $g(x, \widehat{x})$ is minorized by all other minorizing functions of the same form, it is the best possible minorizing function of

the form $ax^k + bx + c$ for given k.

- In the former case we use the fact that at the second supporting point $g(\overline{x_k}, \hat{x}) = f(\overline{x_k})$. Because $g(x, \hat{x})$ minorizes $h(x, \hat{x})$, and they are only equal at $\hat{x}$, $h(\overline{x_k}, \hat{x})$ is larger than $g(\overline{x_k}, \hat{x})$. Now, we obtain that $h(\overline{x_k}, \hat{x}) > g(\overline{x_k}, \hat{x}) = f(\overline{x_k})$, and therefore $h(x, \hat{x})$ is no minorizing function of $f(x)$.
- In the latter case we use the fact that $g(x, \hat{x})$ minorizes $f(x)$, and $h(x, \hat{x})$ minorizes $g(x, \hat{x})$ with unique supporting point $\hat{x}$. Because at all other points, $h(x, \hat{x})$ lies beneath $g(x, \hat{x})$, it also lies beneath $f(x)$ at other points than $\hat{x}$.

□

Elementary Symmetric Functions

The elementary symmetric functions for one element of $\mathbf{b}$ are equal to

$$\gamma_0(\mathbf{b}) = 1,$$

$$\gamma_1(\mathbf{b}) = b_1.$$

For notational convenience, we leave the $(\mathbf{b})$. When we add the element b_2, we also get a new γ, called γ_2. The value of γ_1 also changes. The new values can be computed recursively as follows:

$$\gamma_j = \gamma_j + \gamma_{j-1} b_k.$$

This recursion starts here with γ_2 and ends with γ_1. The outcome is

$$\gamma_0 = 1,$$

$$\gamma_1 = b_1 + b_2,$$

$$\gamma_2 = b_1 b_2.$$

Adding b_3 leads to

$$\begin{aligned}
\gamma_0 &= 1, \\
\gamma_1 &= b_1 + b_2 + b_3, \\
\gamma_2 &= b_1 b_2 + b_1 b_3 + b_2 b_3, \\
\gamma_3 &= b_1 b_2 b_3.
\end{aligned}$$

When one adds new elements b_k, the same procedure is executed. In case one needs to calculate a new value, and does not want to start at the end, the recursion generalizes to:

$$\gamma_j = \gamma_j^{(k)} + \gamma_{j-1}^{(k)} b_k, \tag{17}$$

where $\gamma_j^{(k)}$ is used to denote the j-th elementary symmetric function of $\mathbf{b}$ without the k-th element. One refers to the values of formula 17 with $\gamma_0^{[1]}$ and $\gamma_1^{[1]}$. $\gamma_j^{[k]}$ denotes the j-th elementary symmetric function until element b_k. The elementary symmetric functions stem from the following generalization of the Binomial Theorem:

Theorem 4.

$$\prod_{i=1}^{N}(1 + b_i t) = \sum_{j=0}^{N} \gamma_j t^j.$$

Proof. Assume that $\gamma_j^{[k]}$ equals zero for $j < 0$ and $j > k$. Using (17) in combination with:

$$\begin{aligned}
(1 + b_1 t)(1 + b_2 t) &= (\gamma_0^{[1]} + \gamma_1^{[1]} t)(1 + b_2 t) \\
&= \gamma_0^{[1]} + (\gamma_1^{[1]} + \gamma_0^{[1]} b_2) t + \gamma_1^{[1]} b_2 t^2) \\
&= \gamma_0^{[2]} + \gamma_1^{[2]} t + \gamma_2^{[2]} t^2,
\end{aligned}$$

we obtain:

$$\begin{aligned}\prod_{i=1}^{k}(1+b_i t) &= \left[\prod_{i=1}^{k-1}(1+b_i t)\right](1+b_k t)\\ &= \left[\sum_{j=0}^{k-1}\gamma_j^{[k-1]}t^j\right](1+b_k t)\\ &= \sum_{j=0}^{k-1}\gamma_j^{[k-1]}t^j + \sum_{j=0}^{k-1}\gamma_j^{[k-1]}b_k t^{j+1}\\ &= \sum_{j=0}^{k}\left[\gamma_j^{[k-1]} + \gamma_{j-1}^{[k-1]}b_k\right]t^j = \sum_{j=0}^{k}\left[\gamma_j^{[k]}\right]t^j,\end{aligned}$$

Observe that in general $\gamma_N^{[k]} = \Pi_{i=1}^{k} b_i$ and $\gamma_0^{[k]} = 1$. □

If all k arguments of the elementary symmetric functions are equal to 1 we find that the elementary symmetric functions are the binomial coefficients:

$$\gamma_s(1,1,\ldots,1) = \binom{k}{s}.$$

This result, of course, is hardly surprising since in that case Theorem 4 reduces to the Binomial Theorem.

It is readily found that differentiating an elementary symmetric function with respect to one of its arguments has the following effect:

$$\frac{\partial}{\partial b_j}\gamma_s = \gamma_{s-1}^{(j)}.$$

In Verhelst and Verstralen (1997) the following theorem and corollary are proved:

Theorem 5. *The logarithm of the elementary symmetric functions of k positive arguments $b_1,\ldots,b_k$, considered as a function of their order is strictly concave, i.e.,*

$$\ln(\gamma_s) > \frac{\ln(\gamma_{s-1}) + \ln(\gamma_{s+1})}{2} \quad , s = 1,\ldots,k-1.$$

Corollary 5. *The elementary symmetric functions of k positive arguments* $b_1, \ldots, b_k$, *considered as a function of their order, are single peaked in the following sense:*

1. $\gamma_s \leq \gamma_{s+1} \Rightarrow \gamma_{s-1} < \gamma_s$, $0 < s < k$;
2. $\gamma_s \leq \gamma_{s-1} \Rightarrow \gamma_{s+1} < \gamma_s$, $0 < s < k$.

The following Corollary of Theorem 5 is easily obtained:

Corollary 6. *The ratio of two elementary symmetric functions of k positive arguments* $b_1, \ldots, b_k$ *of consecutive order* $s-1$ *and* s (γ_{s-1}/γ_s) *is strictly increasing in* s.

Proof. The proof is obtained from Theorem 5 as follows: First, observe that the result from Theorem 5 can also be formulated as follows

$$\gamma_s^2 > \gamma_{s-1}\gamma_{s+1} \quad , 0 < s < k.$$

Second, this formulation is equivalent to saying that the ratio γ_{s-1}/γ_s is increasing:

$$\frac{\gamma_{s-1}}{\gamma_s} < \frac{\gamma_s}{\gamma_{s+1}} \Leftrightarrow \gamma_{s-1} < \frac{\gamma_s^2}{\gamma_{s+1}} \Leftrightarrow \gamma_{s-1}\gamma_{s+1} < \gamma_s^2.$$

□

TABLE 9.
Speed of the algorithms.

items	100	100	100	20	60	100	100	100	100	Avg.
persons	100	1000	10000	1000	1000	1000	1000	1000	1000	
δ	$\delta=0$	$\delta=0$	$\delta=0$	$\delta=0$	$\delta=0$	$\delta=1$	$\delta_1=0$ $\delta=-1$	$\delta_1=0$ $\delta=1$	$\delta_1=0$ δ_a [2]	
# fastest										
Impl eq	11	4	5	19	8	92	0	100	4	27
Aitken	1	0	56	48	42	0	100	0	96	38.11
Quadratic	0	0	0	0	0	0	0	0	0	0
False Pos	88	96	39	14	50	6	0	0	0	3.62
Tent	0	0	0	0	0	0	0	0	0	0
1-dim NR	0	0	0	19	0	2	0	0	0	0.26
Avg. time										
Impl eq	1.777	1.856	1.794	0.027	0.457	1.054	5.750	2.120	2.854	1.965
Aitken	1.510	1.467	1.430	0.020	0.361	1.224	2.262	2.267	2.046	1.399
Quadratic	4.990	4.402	4.101	0.044	1.000	4.474	9.281	8.849	7.865	5.001
False Pos	1.389	1.396	1.433	0.023	0.359	1.185	2.352	2.353	2.135	1.403
Tent	6.281	7.533	9.255	0.118	1.947	6.083	11.512	11.465	10.342	7.171
1-dim NR	2.005	1.820	1.699	0.020	0.416	1.559	2.874	2.876	2.550	1.758
M-dim NR	0.752	0.869	0.632	0.009	0.219	0.842	0.730	0.748	0.937	0.638

[2] δ_a: 49 $\delta=-1$, 50 $\delta=1$

TABLE 10.
Number of iterations of the algorithms.

items	100	100	100	20	60	100	100	100	100	Avg.
persons	100	1000	10000	1000	1000	1000	1000	1000	1000	
δ	$\delta=0$	$\delta=0$	$\delta=0$	$\delta=0$	$\delta=0$	$\delta=1$	$\delta_1=0$ $\delta=-1$	$\delta_1=0$ $\delta=1$	$\delta_1=0$ δ_a [3]	
# least its.										
Impl eq	4	3	0	1	3	4	0	0	0	1.67
Aitken	70	94	81	94	94	40	87	76	0	70.67
Quadratic	2	0	1	2	0	0	0	0	0	0.56
False Pos	13	2	2	3	2	2	0	0	2	2.89
Tent	3	1	16	0	0	14	13	24	17	9.78
1-dim N-R	8	0	0	0	1	40	0	0	81	14.44
Avg. # its.										
Impl eq	420.06	422.73	398.32	110.40	269.41	242.31	1333.58	491.89	647.66	481.82
Aitken	185.08	167.18	155.76	40.29	104.78	141.48	271.82	272.05	242.49	175.66
Quadratic	188.69	169.24	157.60	42.70	106.90	171.98	334.99	335.29	293.85	200.14
False Pos	184.45	167.18	155.74	40.30	104.79	141.53	271.83	272.05	241.98	175.54
Tent	217.12	167.28	155.58	40.31	105.37	141.31	272.28	271.86	242.17	179.25
1-dim N-R	185.08	167.20	155.74	40.29	104.78	143.36	272.26	272.42	240.73	175.76
M-dim N-R	3.00	3.00	2.00	2.46	3.00	3.00	3.22	3.30	4.00	3.00

[3] δ_a: 49 $\delta=-1$, 50 $\delta=1$

TABLE 11.

Results from real datasets.

# items	15	40	12
# persons	300	2197	3262
Time			
Impl eq	0.111	0.591	0.070
Aitken	0.040	0.240	0.010
Quadratic	0.090	0.722	0.020
False Pos	0.020	0.220	0.010
Tent	0.100	1.362	0.060
1-dim NR	0.020	0.190	0.000
M-dim NR	0.000	0.090	0.010
# its.			
Impl eq	360	635	347
Aitken	69	104	42
Quadratic	118	155	65
False Pos	64	107	40
Tent	65	105	40
1-dim NR	72	113	35
M-dim NR	4	4	5