Numerical studies for the Rasch model with many items

Strasser, Helmut

DOI:
10.57938/f62a287c-8f4b-4e11-bcd9-6c8cda7bdcd8

Published: 01/10/2012

Numerical Studies for the Rasch Model with Many Items

Helmut Strasser

Research Report Series
Report 119, März 2012

Institute for Statistics and Mathematics
http://statmath.wu.ac.at/

Numerical studies for the Rasch model with many items

Helmut Strasser

March 12, 2012

Abstract

This paper is concerned with numerical studies on the theoretical results obtained in Strasser [1] and [2]. These papers provide asymptotic expansions for conditional expectations of non i.i.d. Bernoulli trials and their application to the covariance structure of conditional maximum likelihood estimates for the Rasch model.

In the present paper systematic numerical studies of the accuracy of the approximations given in Strasser [1] and [2] are presented. It is shown that the order of approximation claimed by the theoretical results can be established numerically.

Contents

1 Introduction

For an introduction into the problem we refer to the introductory sections of the papers by Strasser, [1] and [2]. The numerical studies in the present paper are coded in the S-language. The algorithms can be carried out by the software R. In the appendix of this paper (section 7) we collect some R-code of basic algorithms.

2 Exact calculation of conditional moments

Let $X_1, X_2, \ldots, X_n$ independent random variables with values 0 and 1 and probabilities $P(X_i = 1) = p_i$. Let $S_n := X_1 + X_2 + \cdots + X_n$ be their sum. The main results of Strasser [1] are asymptotic expansions being valid as $n \to \infty$ for the conditional expectations $E(X_i|S_n)$ and $E(X_iX_j|S_n)$ as well as for the expectations of the conditional covariances.

The elementary symmetric polynomial of order s of a vector $\boldsymbol{x} = (x_1, \ldots, x_n)$ is defined by

$$\gamma_{ns} := \sum \Big\{ \prod_{i=1}^{n} x_i^{y_i} : \sum_{i=1}^{n} y_i = s, y_i \in \mathbb{N}_0 \Big\}$$

It is convenient to calculate these polynomials by an algorithm that is similar to Pascal's triangle for binomial coefficients. For elementary symmetric polynomials this algorithm runs as follows:

$$\begin{aligned} &\gamma_{k0} = 1 \text{ whenever } k \geq 0, \\ &\gamma_{kr} = x_k \gamma_{k-1,r-1} + \gamma_{k-1,r} \text{ if } k \geq 1 \text{ and } 1 \leq r \leq k, \\ &\gamma_{kr} = 0, \text{ if } r > k. \end{aligned} \tag{1}$$

The values of the elementary symmetric polynomials increase with n and quickly attain huge values, similarly as binomial coefficients do. For numerical purposes it is therefore preferable to normalize the values of the elementary symmetric polynomials somehow.

There is a relation between elementary symmetric polynomials and certain probabilities which extends the familiar relation between binomial coefficients and binomial probabilities. We have

$$P(S_n = s) = \sum_{\boldsymbol{y}: \sum y_i = s} \prod_i p_i^{y_i} (1 - p_i)^{1 - y_i} = \frac{\gamma_{ns}}{(1 + x_i)^n},$$

where $x_i = p_i/(1 - p_i)$. It is numerically more efficient to calculate these probabilites instead of the elementary symmetric polynomials. It is straightforward to extend the algorithm (1) to the calculation of those probabilities. This algorithm is coded by the R-function (see section 7)

```
prob <- function(p,drop=TRUE)
```

which returns the values of all probabilities $P(S_n = s)$, $s = 0, 1, \ldots, n$, for a given vector $\boldsymbol{p} = (p_i)$ of probabilities. Note that for vectors $\boldsymbol{p}$ with equal components binomial probabilities are obtained.

2.1 EXAMPLE.

```
> prob(rep(0.3,10))
 [1] 0.0282475249 0.1210608210 0.2334744405 ...

> prob((1:9)/10)
 [1] 0.00036288 0.00699984 0.04820760 0.15974936 ...
```

After these preparations it is easy to calculate conditional expectations. Let

$$S_{n-1}^{i} := \sum_{k \neq i} X_k, \quad S_{n-2}^{ij} := \sum_{k \neq i,j} X_k$$

Then we have

$$E(X_i|S_n = s) = \frac{P(X_i = 1,\, S_n = s)}{P(S_n = s)} = p_i \frac{P(S_{n-1}^{i} = s - 1)}{P(S_n = s)} \tag{2}$$

$$E(X_i X_j|S_n = s) = \frac{P(X_i = 1,\, X_j = 1,\, S_n = s)}{P(S_n = s)} = p_i p_j \frac{P(S_{n-2}^{ij} = s - 2)}{P(S_n = s)},\ i \neq j. \tag{3}$$

These formulas are coded by the R-function (see section 7)

```
cm_exact <- function(p,i=1,j=NULL)
```

where $\boldsymbol{p}$ is a vector of probabilities. The function returns vectors $a_s(i) = E(X_i|S_n = s)$ and $b_s(i,j) = E(X_i X_j|S_n = s)$. Note that for vectors $\boldsymbol{p}$ with equal components there are explicit expressions for the conditional moments:

$$a_s(i) = E(X_i|S_n = s) = \frac{s}{n}$$

$$b_s(i,j) = E(X_i X_j|S_n = s) = \frac{s(s-1)}{n(n-1)},\ i \neq j.$$

2.2 EXAMPLE.

```
> cm_exact(rep(0.1,5),1)
[1] 0.0 0.2 0.4 0.6 0.8 1.0

> cm_exact(rep(0.1,5),1,2)
```

```
[1] 0.0 0.0 0.1 0.3 0.6 1.0

> cm_exact(runif(5),1)
[1] 0.0000000 0.2049964 0.4767453 0.7231139 0.9178962 ...

> cm_exact(runif(5),1,2)
[1] 0.00000000 0.00000000 0.04486523 0.21934904 0.65105522
[6] 1.00000000
```

Next we would like to calculate the conditional covariance matrix

$$\boldsymbol{F}_{ns} := E(\boldsymbol{X}\boldsymbol{X}^t|S_n = s) - E(\boldsymbol{X}|S_n = s)E(\boldsymbol{X}|S_n = s)^t \tag{4}$$

and the expectation of the conditional covariance matrix

$$\boldsymbol{F}_n := E\Big(E(\boldsymbol{X}\boldsymbol{X}^t|S_n) - E(\boldsymbol{X}|S_n)E(\boldsymbol{X}|S_n)^t\Big) = \sum_{s=0}^{n} \boldsymbol{F}_{ns}P(S_n = s) \tag{5}$$

These evaluations are obviously based on the previous calculations. They are coded by the R-functions (see section 7)

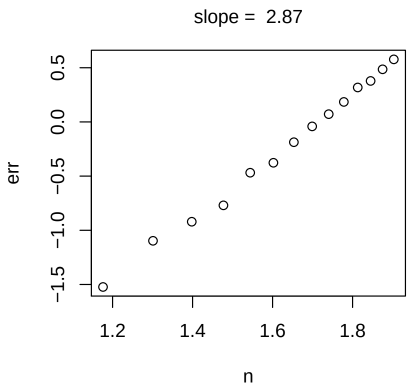
```
vcond_exact <- function(p)
v_exact <- function(p)
```

The function `vcond_exact` simply collects the results of `cm_exact` and puts them into an array $c_{ijs} = (\boldsymbol{F}_{ns})_{ij}$ according to (4). The function `v_exact` evaluates (5) and returns the matrix $\boldsymbol{F}_n$.

It should be noted that the evaluation of these matrices for large n needs computing time.

2.3 EXAMPLE. Let us carry out a computer experiment concerning the calculation time of $\boldsymbol{F}_n$ for increasing n. The R-function `errplot` (see section 7) draws a log-log plot and calculates the slope of the regression line.

```
> p=runif(50)
> tm=numeric(0)
> nn=seq(10,80,by=5)
> for (n in nn) tm=c(tm,system.time(v_exact(runif(n)))[1])
> errplot(nn[-1],tm[-1])
```

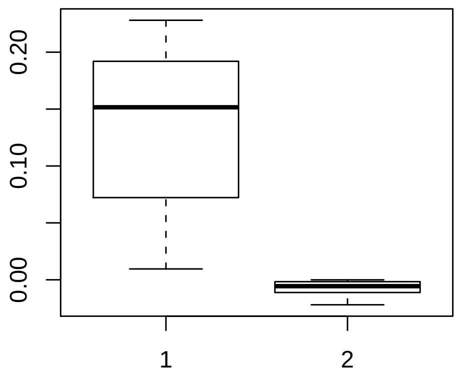


The log-log plot indicates an increase of computing time of order $n^{2.87}$. It is, however, to be expected that for large values of n the increase of computing time will turn to exponential order. The reason is that the function `prob` will finally dominate the complexity of the algorithm.

It should be noted that the matrix $\boldsymbol{F}_n$ typically has a diagonal structure.

2.4 EXAMPLE.

```
> x=v_exact(runif(20))
> y=diag(x)
> z=x[lower.tri(x)]
> boxplot(list(y,z))
```



3 Approximation of conditional moments

Let

$$\overline{p} := \frac{1}{n}\sum_{j=1}^{n} p_j, \quad v_j := p_j(1-p_j), \quad \sigma_n^2 := \sum_{j=1}^{n} v_j$$

and

$$\tau_{ni} := 2\Big(p_i - \sum_{j=1}^{n} \frac{v_j}{\sigma_n^2} p_j\Big).$$

In Strasser [1] approximations for the conditional moments (2) and (3) are given in terms of polynomials of

$$Z_n := \frac{S_n - n\overline{p}}{\sigma_n}.$$

The approximation of $E(X_i|S_n)$ according to Theorem 2.1 in Strasser [1] is given by

$$E(X_i|S_n) \approx p_i + \frac{1}{\sqrt{n}}\frac{v_i}{\overline{\sigma}_n} Z_n - \frac{1}{n}\frac{v_i\tau_{ni}}{2\overline{\sigma}_n^2}(Z_n^2 - 1) \tag{6}$$

Note that the linear part of the approximation is the linear regression function of X_i with respect to S_n.

The approximation of $E(X_iX_j|S_n)$ according to Lemma 2.2 in Strasser [1] for $i \neq j$ is given by

$$\begin{aligned} E(X_iX_j|S_n) \approx p_ip_j + \frac{1}{\sqrt{n}}\frac{p_iv_j + p_jv_i}{\overline{\sigma}_n} Z_n \\ + \frac{1}{n}\frac{v_iv_j}{\overline{\sigma}_n^2}(Z_n^2 - 1) - \frac{1}{n}\frac{p_iv_j\tau_{nj} + p_jv_i\tau_{ni}}{2\overline{\sigma}_n^2}(Z_n^2 - 1) \end{aligned} \tag{7}$$

Both approximations have a theoretical error term of order $O(n^{-3/2})$. The approximations are uniform as long as (Z_n) is uniformly bounded. In view of the central limit theorem the sequence of random variables (Z_n) is uniformly bounded with probabilities arbitrarily near to one, i.e.

$$\sup_n P(|Z_n| \geq a_\epsilon) < \epsilon \text{ for every } \epsilon > 0 \text{ and suitable } a_\epsilon < \infty.$$

In this sense the approximations are valid with probabilities arbitrary close to one.

The approximation polynomials (6) and (7) are coded by the R-function (see section 7)

```
cm_approx <- function(p,i=1,j=NULL)
```

Let us carry out some numerical experiments. We start with a simple illustration.

3.1 EXAMPLE.

```
> p=runif(30)
> x=cm_exact(p,1)
> y=cm_approx(p,1)
> summary(y-x)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.340600 -0.095110 -0.002954 -0.051520  0.001969  0.079250

> x=cm_exact(p,1,2)
> y=cm_approx(p,1,2)
> summary(y-x)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.2740000 -0.0644400 -0.0009374 -0.0376500  0.0106400  0.1258000
```

The error distributions have heavy tails. This is due to the fact that the approximation is only uniform on domains where (Z_n) is bounded.

Now we are going to study a sequence of approximations based on a randomly chosen vector $\boldsymbol{p}$ of length n. We compute the maximal absolute error between the exact values and the approximate values for all vectors $\boldsymbol{p}_k = (p_1, \ldots, p_k)$, $k = 1, 2, \ldots, n$. For this we apply the R-function

```
test1 <- function(nn,i=1,j=NULL)
```

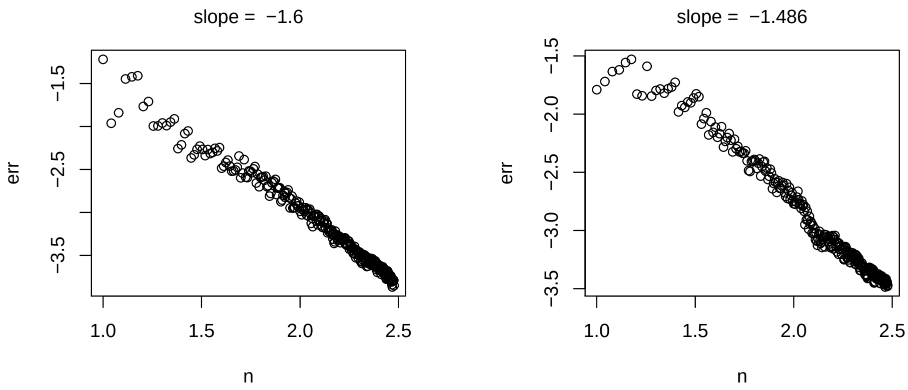
The function `test1` calculates the maximal absolute errors of the approximation polynomials (6) and (7) on the range $|Z_n| < N_{0.995}$ (where N_α denotes the α-quantile of the normal distribution).

When we consider the log-log plots (log of base 10) of the maximal absolute errors with respect to the length of $\boldsymbol{p}$ then we observe a linear pattern which indicates a power function relation. The log-log plot also displays the slope of the regression line. Recall that the theoretical results predict a slope of 1.5.

3.2 EXAMPLE. We choose $n = 300$.

```
> res=test1(300,1)
> errplot(res$n,res$err)

> res=test1(300,1,2)
> errplot(res$n,res$err)
```



These numerical results support the quality of the theoretical error bounds.

The next step is the approximation of the conditional covariance matrix $\boldsymbol{F}_{ns}$ based on the preceding approximations of the conditional moments. This coded by the R-function (see section 7)

```
vcond_approx <- function(p)
```

This function simply collects the results of `cm_approx` and puts them into an array. The quality of approximation is the same as for the single components.

Things are becoming much more interesting when we pass to the expectation $\boldsymbol{F}_n$ of the conditional covariance matrix $\boldsymbol{F}_{ns}$. Theorem 2.4 in Strasser [1] shows that $\boldsymbol{F}_n$ can be approximated with very simple expressions. These expressions are far simpler than those which we would obtain by proceeding in a similar way as in (5). The approximations are of a considerably higher order and are valid in a stronger sense.

The matrix norm of a positive semidefinite symmetric matrix is defined by

$$||\boldsymbol{A}|| = \sup\{\boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} : ||\boldsymbol{x}|| = 1\}, \tag{8}$$

(see the R-function `nm` in 7.) Let

$$\boldsymbol{G}_{n,ij} := v_i\delta_{ij} - \frac{v_i v_j}{\sigma_n^2} \tag{9}$$

and

$$\boldsymbol{H}_{n,ij} := \boldsymbol{G}_{n,ij} + \frac{v_i v_j \tau_i \tau_j}{2\sigma_n^4} \tag{10}$$

Then it is shown in Theorem 2.4 of Strasser [1] and in Corollary 2.2 of Strasser [2] that

$$\boldsymbol{F}_n = \boldsymbol{G}_n + O(n^{-2}) \text{ and } ||\boldsymbol{F}_n - \boldsymbol{G}_n|| = O(n^{-1}) \tag{11}$$

and

$$\boldsymbol{F}_n = \boldsymbol{H}_n + O(n^{-5/2}) \text{ and } ||\boldsymbol{F}_n - \boldsymbol{H}_n|| = O(n^{-3/2}) \tag{12}$$

The matrices $\boldsymbol{G}_n$ and $\boldsymbol{H}_n$ are coded by the R-functions (see section 7)

```
v_approx <- function(p, linear=FALSE)
```

The option `linear` returns $\boldsymbol{G}_n$, otherwise $\boldsymbol{H}_n$ is returned. Let us consider some numeric examples.

3.3 EXAMPLE. We randomly choose 100 vectors $\boldsymbol{p}$ of length $n = 20$ and compare $\boldsymbol{F}_n$ with $\boldsymbol{G}_n$ and $\boldsymbol{H}_n$, respectively.

```
x=numeric(0)
y=x
z=x
u=x
for (i in 1:100) {
   p=runif(20)
   a=v_exact(p)
   b=v_approx(p, linear=TRUE)
   c=v_approx(p, linear=FALSE)
   x=c(x, max(abs(a-b)))
   y=c(y, max(abs(a-c)))
   z=c(z, nm(a-b))
   u=c(z, nm(a-c))
   }

> summary(x)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0003435 0.0004523 0.0005354 0.0005738 0.0006347 0.0015420
> summary(y)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
2.133e-05 3.531e-05 4.502e-05 5.468e-05 6.122e-05 2.304e-04
> summary(z)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.001845 0.003498 0.004385 0.004427 0.005044 0.008277
> summary(u)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.0002906 0.0034340 0.0043570 0.0043860 0.0050170 0.0082770
```

Next we study a sequence of approximations based on a randomly chosen vector $\boldsymbol{p}$ of length n. We compute the maximal absolute error and the norm distances between the exact values and the approximate values for all vectors $\boldsymbol{p}_k = (p_1, \ldots, p_k)$, $k = 1, 2, \ldots, n$. For this we apply the R-function

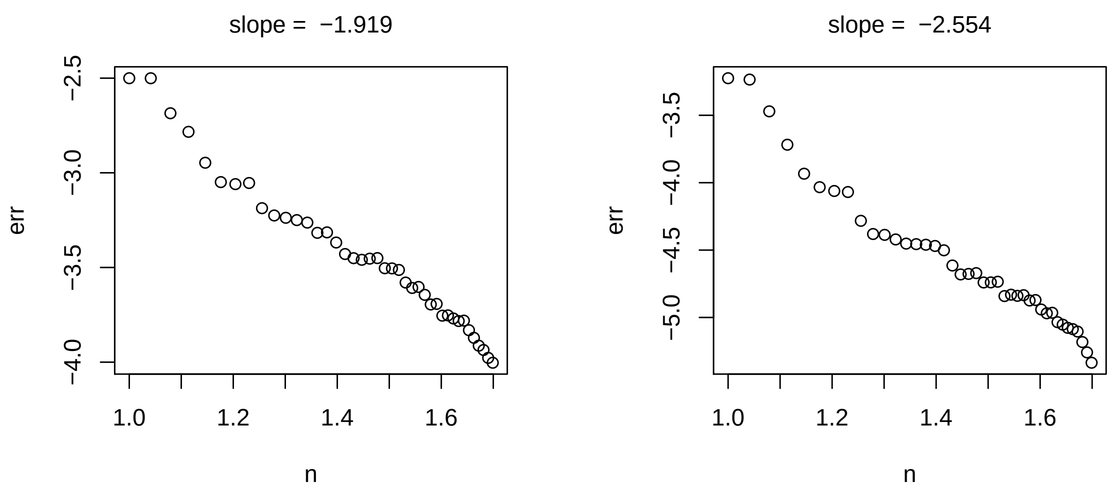
```
test2 <- function(nn)
```

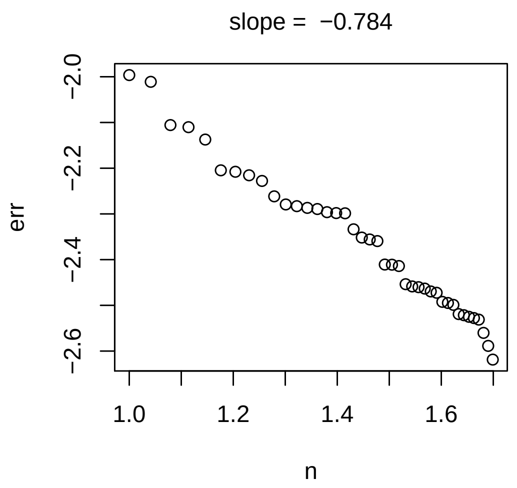
The function `test2` calculates the errors of the approximated matrices (9) and (10). When we consider the log-log plots (log of base 10) then we observe a linear pattern which indicates a power function relation. The slope of the log-log plot estimates the exponent of the order of decrease.

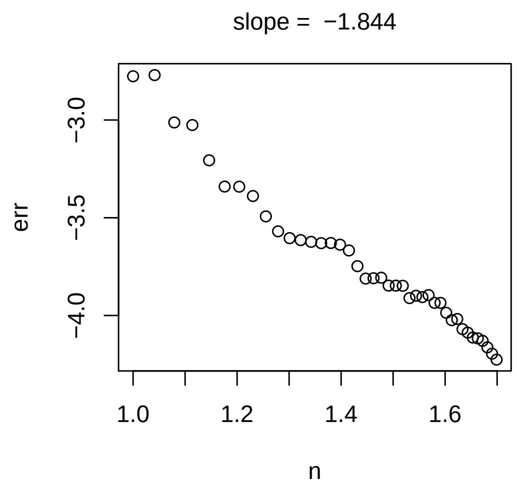
3.4 EXAMPLE. We choose $n = 50$.

```
res=test2(50)

errplot(res$n, res$g_abs)
errplot(res$n, res$h_abs)
errplot(res$n, res$g_nm)
errplot(res$n, res$h_nm)
```





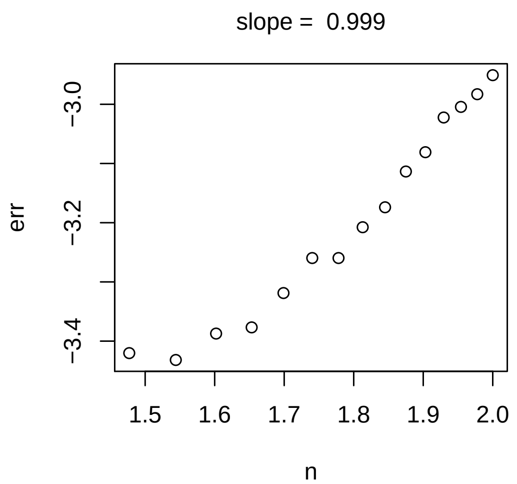


First two plots show the maximal absolute errors between on the one hand $\boldsymbol{F}_n$ and on the other hand $\boldsymbol{G}_n$ and $\boldsymbol{H}_n$, respectively. The other two plots show the matrix norm of the errors. The plots illustrate the validity of the theoretical results.

Finally, let us carry out a computer experiment concerning the calculation time of $\boldsymbol{F}_n$ for increasing n.

3.5 EXAMPLE.

```
> p=runif(50)
> tm=numeric(0)
> nn=seq(30,100,by=5)
> fun = function(n) for (i in 1:1000) v_approx(runif(n))
> for (n in nn) tm=c(tm,system.time(fun(n))[1]/1000)
> errplot(nn,tm)
```



The log-log plot indicates an increase of computing time of linear order n. This is a complexity of almost 2 orders less than for the computation of the exact matrices. Apart from complexity the absolute speed level is considerably smaller.

4 Conditional m.l. estmation for the Rasch model

The Rasch model is a parametric statistical model for the distribution of von n Bernoulli random variables, where the parameter vector $\boldsymbol{\theta}$ defines the probabilities by

$$p_i = \frac{e^{\theta_i}}{1 + e^{\theta_i}}$$

If we denote the elementary symmetric functions of the vector $\epsilon_i = e^{\theta_i}$ by γ_{ns} then we have

$$P_{\boldsymbol{\theta}}(S_n = s) = \sum_{\boldsymbol{y}:\sum y_i = s} \prod_i \frac{\epsilon_i^{y_i}}{1 + \epsilon_i} = \frac{\gamma_{ns}}{\prod_i (1 + \epsilon_i)}$$

The conditional probabilities given $S_n = s$ are expressions of the form

$$q(x, \boldsymbol{\theta}) := P_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x} | S_n = s) = \frac{e^{\sum_i \theta_i x_i}}{\gamma_{ns}} \text{ whenever } s = s_n(\boldsymbol{x}).$$

It is easy to see that these conditional probabilities do not depend on the mean value $\overline{\boldsymbol{\theta}}$ but only on the deviations of the components of $\boldsymbol{\theta}$ from the mean value $\overline{\boldsymbol{\theta}}$. In other words, the conditional probabilities are invariant with respect to translations of $\boldsymbol{\theta}$ in direction $\boldsymbol{e} = (1, 1, \ldots, 1)$.

If we consider the parameter to be a sum $\boldsymbol{\theta} = \boldsymbol{\beta} + \tau$ of a vector $\boldsymbol{\beta}$ of structural parameters and a one-dimensional nuisance parameter τ, then this invariance property makes it reasonable to use the conditional loglikelihoods as contrast functions for the estimation of the structural parameters under incidental nuisance parameters. This leads to conditional maximum likelihood estimation.

The conditional loglikelihoods are

$$\ell(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_i \theta_i x_i - \log \gamma_{n,s_n(\boldsymbol{x})}$$

and hence, for the the partial derivatives we obtain

$$\frac{\partial}{\partial \theta_i} \ell(\boldsymbol{x}, \boldsymbol{\theta}) = -x_i + E_{\boldsymbol{\theta}}(X_i | S_n = s_n(\boldsymbol{x}))$$

since the conditional expectation of the partial derivatives has to be zero. As a consequence the conditional Fisher information turns out to be the conditional covariance matrix

$$\begin{aligned} \boldsymbol{F}_n(\boldsymbol{\theta}, s) &= E_{\boldsymbol{\theta}}(D_1 \ell_{\boldsymbol{\theta}} \otimes D_1 \ell_{\boldsymbol{\theta}} | S_n = s) \\ &= E_{\boldsymbol{\theta}}(\boldsymbol{X}\boldsymbol{X}^t | S_n = s) - E_{\boldsymbol{\theta}}(\boldsymbol{X} | S_n = s) E_{\boldsymbol{\theta}}(\boldsymbol{X} | S_n = s)^t \end{aligned} \tag{13}$$

of the observation vector $\boldsymbol{X}$ given $S_n = s$.

It should be noted that by translation invariance of the conditional distributions the partial derivatives of the loglikelihoods are orthogonal to $\boldsymbol{e}$, which means that the row- and column-sums of $\boldsymbol{F}_n(\boldsymbol{\theta}, s)$ is zero.

The unconditional Fisher information of conditional maximum likelihood estimation for the i.i.d. case (identical nuisance parameters) is given by

$$\boldsymbol{F}_n(\boldsymbol{\theta}) := E_{\boldsymbol{\theta}}(D_1 \ell_{\boldsymbol{\theta}} \otimes D_1 \ell_{\boldsymbol{\theta}}), \tag{14}$$

If the nuisance parameters follow a distribution Γ then the Fisher information

$$\boldsymbol{F}_n(\boldsymbol{\theta}, \Gamma) := \int E_{\boldsymbol{\theta}+\tau}(D_1 \ell_{\boldsymbol{\theta}+\tau} \otimes D_1 \ell_{\boldsymbol{\theta}+\tau}) \, \Gamma(d\tau)$$

By Lemma 3.2 of Strasser [2] the asymptotic covariance matrix of the conditional maximum likelihood estimates is identical to the Moore-Penrose pseudoinverse $\boldsymbol{F}^+_{n\boldsymbol{\beta}\Gamma}$ of the Fisher information.

5 Exact calculations for the Rasch model

Let $\boldsymbol{\theta} = \boldsymbol{\beta} + \tau$ where $\overline{\boldsymbol{\beta}} = 0$.

It is clear form the preceding section 4 that the Fisher information in the i.i.d. case can be calculated as the expectation of the conditional covariance matrix

$$\boldsymbol{F}_{n\boldsymbol{\beta}\tau} := \boldsymbol{F}_n(\boldsymbol{\theta}) = E_{\boldsymbol{\theta}}\Big(E_{\boldsymbol{\theta}}(\boldsymbol{X}\boldsymbol{X}^t|S_n = s) - E_{\boldsymbol{\theta}}(\boldsymbol{X}|S_n = s)E_{\boldsymbol{\theta}}(\boldsymbol{X}|S_n = s)^t\Big).$$

In the case of a random nuisance parameter with distribution Γ we have

$$\boldsymbol{F}_{n\boldsymbol{\beta}\Gamma} := \boldsymbol{F}_n(\boldsymbol{\theta}, \Gamma) = \int \boldsymbol{F}_{n\tau}\,\Gamma(d\tau)$$

The exact calculation of $\boldsymbol{F}_{n\beta\Gamma}$ for an empirical distribution Γ is coded by the R-function

```
f_exact <- function(beta, tau=0)
```

The parameter `tau` contains the data vector for the empirical distribution Γ.

The exact calculation of the asymptotic covariance matrix $\boldsymbol{F}^+_{n\boldsymbol{\beta}\Gamma}$ can be obtained as the Moore-Penrose pseudoinverse which is coded by the R-function

```
pinv <- function(x, eps=1e-10)
```

6 Approximations for the Rasch model

6.1 The Fisher information

The most simple approximation of the Fisher information for the i.i.d. case is given by equation (9) which now is a function of $\boldsymbol{\beta}$ and τ. In the case of a random nuisance parameter the approximation provide by equation (9) is

$$\boldsymbol{G}_{n\boldsymbol{\beta}\Gamma} := \int \boldsymbol{G}_{n\boldsymbol{\beta}\tau}\,\Gamma(d\tau)$$

A more sophisticated approximation is provided by equation (10) leading to

$$\boldsymbol{H}_{n\boldsymbol{\beta}\Gamma} := \int \boldsymbol{H}_{n\boldsymbol{\beta}\tau}\,\Gamma(d\tau)$$

This approximations are coded by the R-function

```
f_approx <- function(beta, tau=0, linear=FALSE)
```

Let us consider some numeric examples.

6.1 EXAMPLE. We randomly choose 100 vectors $\boldsymbol{\beta}$ of length $n = 20$ and a vector of length 20 of nuisance parameters τ. We compare $\boldsymbol{F}_{n\boldsymbol{\beta}\Gamma}$ with $\boldsymbol{G}_{n\boldsymbol{\beta}\Gamma})$ and $\boldsymbol{H}_{n\boldsymbol{\beta}\Gamma})$ where Γ is the empirical distribution the nuisance parameter.

```
> x=numeric(0)
> y=x
> z=x
> u=x
> for (i in 1:100) {
+     beta=rnorm(20)
+     tau=rnorm(20)
+     a=f_exact(beta, tau)
+     b=f_approx(beta, tau, linear=TRUE)
+     c=f_approx(beta, tau, linear=FALSE)
+     x=c(x, max(abs(a-b)))
+     y=c(y, max(abs(a-c)))
+     z=c(z, nm(a-b))
+     u=c(z, nm(a-c))
+     }
> summary(x)
     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
0.0001906  0.0003882  0.0004514  0.0004858  0.0005404  0.0008830
> summary(y)
     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
1.050e-05  3.549e-05  5.092e-05  6.724e-05  8.364e-05  2.819e-04
> summary(z)
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.001226  0.001956  0.002280  0.002323  0.002688  0.003832
> summary(u)
     Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
9.708e-05  1.937e-03  2.254e-03  2.301e-03  2.685e-03  3.832e-03
```

Next we study a sequence of approximations based on a randomly chosen vector $\boldsymbol{\beta}$ of length n. We compute the errors between the exact values and the approximate values for all vectors $\boldsymbol{\beta}_k = (\beta_1, \ldots, \beta_k)$, $k = 1, 2, \ldots, n$. For this we apply the R-function

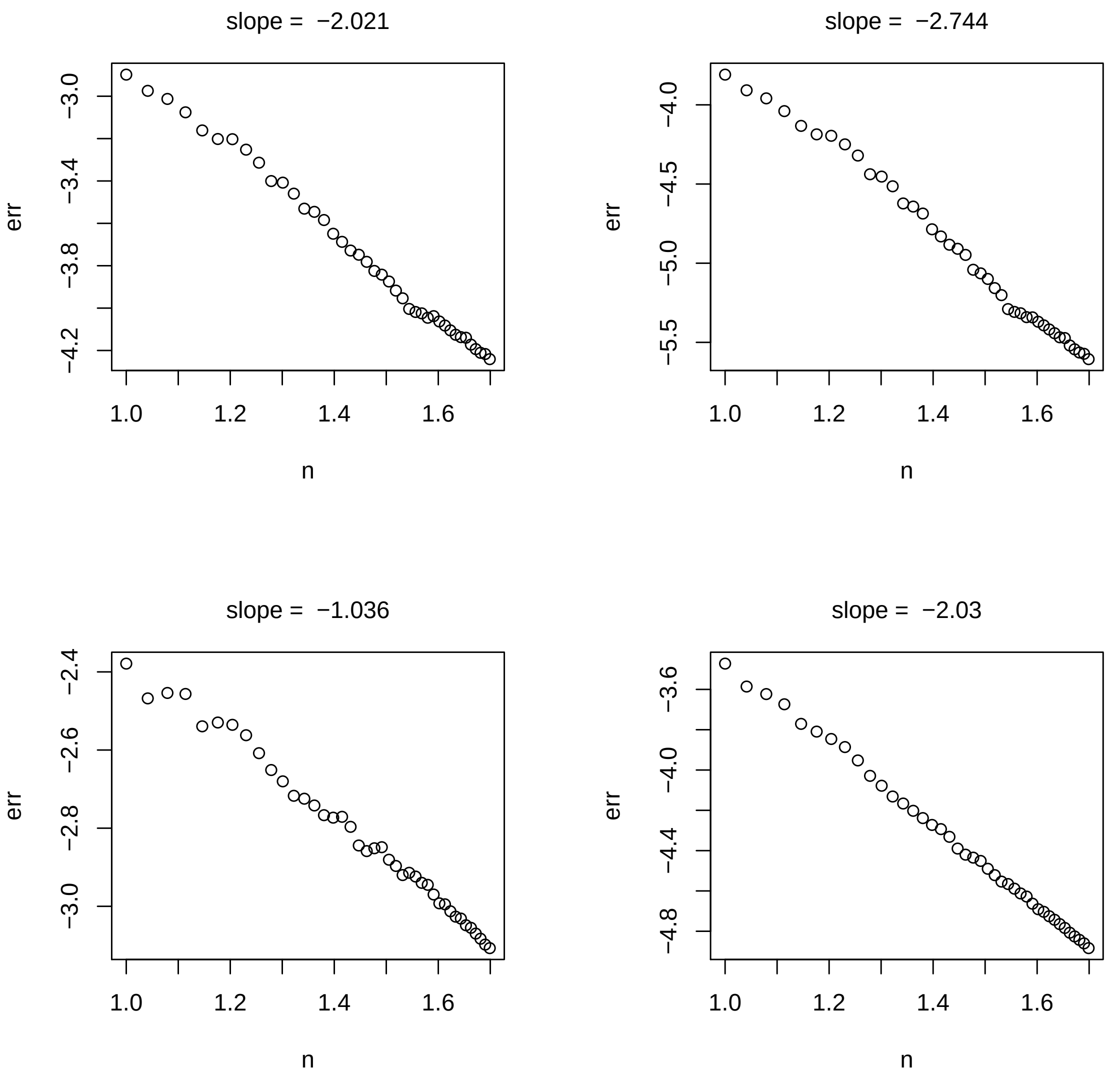
```
test3 <- function(nn)
```

The function `test3` calculates the errors of the approximated matrices (9) and (10). When we consider the log-log plots (log of base 10) then we observe a linear pattern which indicates a power function relation. The slope of the log-log plot estimates the exponent of the order of decrease.

6.2 EXAMPLE. We choose $n = 50$.

```
res=test3(50)

errplot(res$n,res$g_abs)
errplot(res$n,res$h_abs)
errplot(res$n,res$g_nm)
errplot(res$n,res$h_nm)
```



First two plots show the maximal absolute errors between on the one hand $\boldsymbol{F}_{n\beta\Gamma}$ and on the other hand $\boldsymbol{G}_{n\beta\Gamma}$ and $\boldsymbol{H}_{n\beta\Gamma}$, respectively. The other two plots show the matrix norm of the errors. The plots illustrate the validity of the theoretical results.

6.2 The asymptotic covariance matrix

Basically the asymptotic covariance matrix can be approximated by calculating die Moore-Penrose pseudoinverse of the approximate Fisher information. This is proved by Strasser [2] in Theorems 2.3 and 2.4. Moore-Penrose pseudoinverses can be calculated by well-known algorithms (see the R-function `pinv` in section 7).

Let us perform the same numerical experiments as for the Fisher information.

6.3 EXAMPLE. We randomly choose 100 vectors $\boldsymbol{\beta}$ of length $n = 30$ and a vector of length 30 of nuisance parameters τ. We compare $\boldsymbol{F}^+_{n\boldsymbol{\beta}\Gamma}$ with $\boldsymbol{G}^+_{n\boldsymbol{\beta}\Gamma}$ and $\boldsymbol{H}^+_{n\boldsymbol{\beta}\Gamma}$ where Γ is the empirical distribution the nuisance parameter.

```
> x=numeric(0)
> y=x
> z=x
> u=x
> for (i in 1:100) {
+     beta=rnorm(30)
+     tau=rnorm(30)
+     a=pinv(f_exact(beta,tau))
+     b=pinv(f_approx(beta,tau,linear=TRUE))
+     c=pinv(f_approx(beta,tau,linear=FALSE))
+     x=c(x,max(abs(a-b)))
+     y=c(y,max(abs(a-c)))
+     z=c(z,nm(a-b))
+     u=c(z,nm(a-c))
+     }
> summary(x)
   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.004479 0.015140 0.021880 0.030950 0.032700 0.189100
> summary(y)
    Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
0.0001929 0.0010270 0.0020750 0.0045350 0.0040890 0.0519000
> summary(z)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.03998 0.06354 0.08139 0.08703 0.10280 0.23680
> summary(u)
   Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
0.001983 0.062670 0.080620 0.086190 0.102600 0.236800
```

Next we study a sequence of approximations based on a randomly chosen vector $\boldsymbol{\beta}$ of length n. We compute the errors between the exact values and the approximate values for all vectors $\boldsymbol{\beta}_k = (\beta_1, \ldots, \beta_k)$, $k = 1, 2, \ldots, n$. For this we apply the R-function

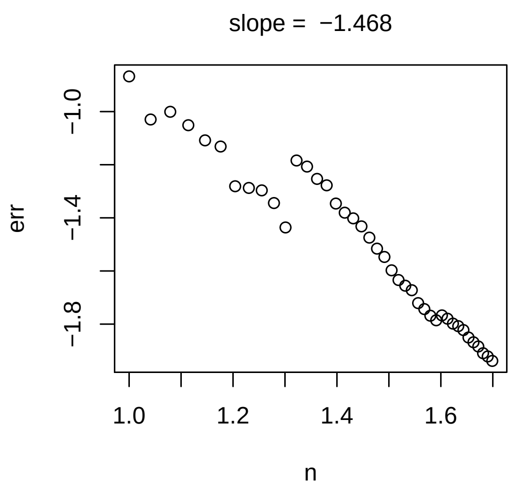
```
test4 <- function(nn)
```

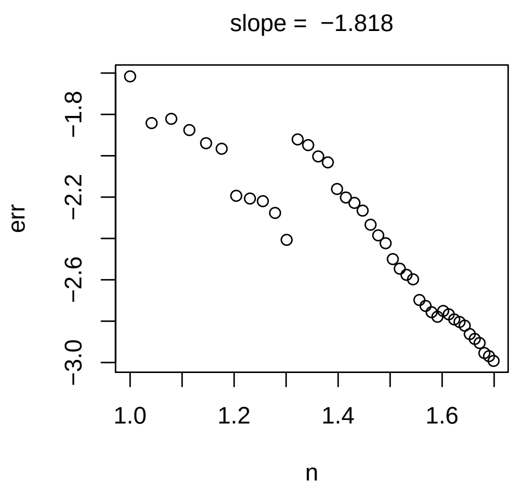
which is a simple adaption of `test3`.

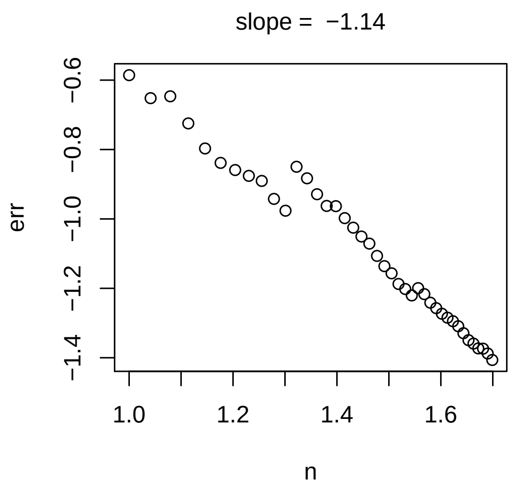
6.4 EXAMPLE. We choose $n = 50$.

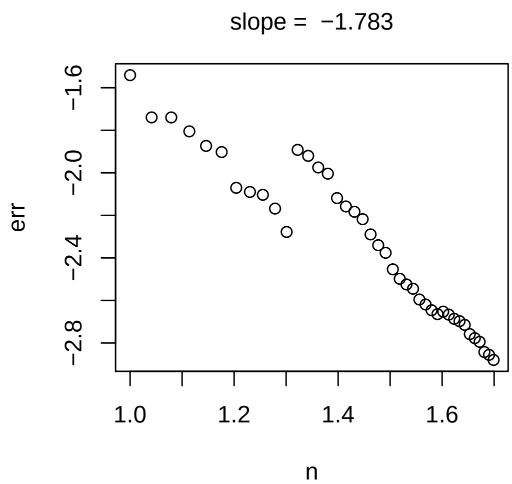
```
res=test4(50)

errplot(res$n, res$g_abs)
errplot(res$n, res$h_abs)
errplot(res$n, res$g_nm)
errplot(res$n, res$h_nm)
```









The first two plots show the maximal absolute errors between on the one hand $\boldsymbol{F}^{+}_{n\beta\Gamma}$ and on the other hand $\boldsymbol{G}^{+}_{n\beta\Gamma}$ and $\boldsymbol{H}^{+}_{n\beta\Gamma}$, respectively. The other two plots show the matrix norm of the errors. The plots illustrate the validity of the theoretical results.

7 Appendix: R-Code

```
prob <- function(p) {
   k=length(p)
   x=p/(1-p)
   a=c(1-p[1],p[1])
   if (k==1) return(a)
   for (r in 2:k) a=(c(0,a)*x[r]+c(a,0))*(1-p[r])
   return(a)
   }

cm_exact <- function(p,i=1,j=NULL) {
   stopifnot(length(p)>1)
   if (is.null(j)||(j==i)) {
     p1=c(0,prob(p[-i]))
     p2=prob(p)
     return(p[i]*p1/p2)
     }
   else {
     p1=c(0,0,prob(p[-c(i,j)]))
     p2=prob(p)
     return(p[i]*p[j]*p1/p2)
     }
```

```
}

vcond_exact <- function(p) {
   k=length(p)
   m=matrix(0,k,k+1)
   f=array(0,c(k,k,k+1))
   for (i in 1:k) {
     m[i,]=cm_exact(p,i)
     f[i,i,]=m[i,]-m[i,]^2
     }
   for (i in 1:(k-1)) for (j in (i+1):k) {
     f[i,j,]=cm_exact(p,i,j)-m[i,]*m[j,]
     f[j,i,]=f[i,j,]
     }
   return(f)
   }

v_exact <- function(p) {
   k=length(p)
   q=prob(p)
   cf=vcond_exact(p)
   f=matrix(0,k,k)
   for (s in 1:(k+1)) f=f+cf[,,s]*q[s]
   return(f)
   }


cm_approx <- function(p,i=1,j=NULL) {
   stopifnot(length(p)>1)
   n=length(p)
   v=p*(1-p)
   ss=sum(v)
   z=(0:n-sum(p))/sqrt(ss)
   tau=2*(p-sum(v*p)/ss)
   if (is.null(j)||(j==i))
     return(p[i]+v[i]*z/sqrt(ss)-v[i]*tau[i]*(z^2-1)/(2*ss))
   else {
     a=p[i]*p[j]
     b=p[i]*v[j]+p[j]*v[i]
     c=v[i]*v[j]
     d=p[i]*v[j]*tau[j]+p[j]*v[i]*tau[i]
     return(a+b*z/sqrt(ss)+(z^2-1)*(c-d/2)/ss)
     }
   }

rng <-function(p,lv=0.99) {
   n=length(p)
   s=0:n
   v=p*(1-p)
```

```
  z=(s-sum(p))/sqrt(sum(v))
  i=which(abs(z)<qnorm(lv+(1-lv)/2))
  return(i)
  }

vcond_approx <- function(p) {
  k=length(p)
  m=matrix(0,k,k+1)
  f=array(0,c(k,k,k+1))
  for (i in 1:k) {
    m[i,]=cm_approx(p,i)
    f[i,i,]=m[i,]-m[i,]^2
    }
  for (i in 1:(k-1)) for (j in (i+1):k) {
    f[i,j,]=cm_approx(p,i,j)-m[i,]*m[j,]
    f[j,i,]=f[i,j,]
    }
  return(f)
  }

test1 <- function(nn,i=1,j=NULL) {
  if (is.null(j)) j=i
  e=numeric(0)
  p0=runif(nn)
  m=max(c(i,j,10))
  nn=max(m,nn)
  for (n in m:nn) {
    p=p0[1:n]
    s=rng(p)
    x=cm_exact(p,i,j)[s]
    y=cm_approx(p,i,j)[s]
    e=c(e,max(abs(x-y)))
    }
  return(list(n=m:nn,err=e))
  }

errplot <- function(n,e) {
  windows(4,4)
  plot(log10(n),log10(e),xlab="n",ylab="err")
  fm=lm(log10(e)~log10(n))
  mtext(paste("slope=", round(fm$coef[2],3)),3,1)
  }

v_approx <- function(p,linear=FALSE){
  v=p*(1-p)
  if (linear) return(diag(v)-outer(v,v)/sum(v))
  else {
    c3=p*(1-p)*(1-2*p)
    w=(2*p-1)/sqrt(mean(v))+mean(c3)/mean(v)^(3/2)
```

```
    return(diag(v)-outer(v,v)/sum(v)*(1+outer(w,w)/2/length(p)))
    }
  }


nm <- function(a) sqrt(max(eigen(t(a)%*%a)$values))

test2 <- function(nn) {
  p0=runif(nn)
  m=10
  nn=max(m,nn)
  x=numeric(0)
  y=x
  z=x
  u=x
  for (n in m:nn) {
    Cat(n)
    p=p0[1:n]
    a=v_exact(p)
    b=v_approx(p,linear=TRUE)
    c=v_approx(p,linear=FALSE)
    x=c(x,max(abs(a-b)))
    y=c(y,max(abs(a-c)))
    z=c(z,nm(a-b))
    u=c(u,nm(a-c))
    }
  return(list(n=m:nn,g_abs=x,h_abs=y,g_nm=z,h_nm=u))
  }

f_exact <- function(beta,tau=0) {
  beta=beta-mean(beta)
  k=length(beta)
  q=numeric(k+1)
  for (i in 1:length(tau))
    q=q+prob(exp(beta+tau[i])/(1+exp(beta+tau[i])))
  q=q/length(tau)
  cf=vcond_exact(exp(beta)/(1+exp(beta)))
  f=matrix(0,k,k)
  for (s in 1:(k+1)) f=f+cf[,,s]*q[s]
  return(f)
  }

pinv <- function(x,eps=1e-10){
  y=svd(x)
  dm=min(dim(x))
  chk=y$d>eps
  d=(1/(y$d+1-chk))*chk
  return(y$v%*%diag(d,dm,dm)%*%t(y$u))
```

```
  }

f_approx  <- function(beta, tau=0, linear=FALSE){
  beta=beta-mean(beta)
  k=length(beta)
  f=matrix(0,k,k)
  for (i in 1:length(tau)) {
    p=exp(beta+tau[i])/(1+exp(beta+tau[i]))
    f=f+v_approx(p, linear=linear)
  f=f/length(tau)
  return(f)
  }

test3 <- function(nn) {
  beta0=rnorm(nn)
  tau0=rnorm(nn)
  m=10
  nn=max(m, nn)
  x=numeric(0)
  y=x
  z=x
  u=x
  for (n in m:nn) {
    Cat(n)
    beta=beta0[1:n]
    tau=tau0[1:n]
    a=f_exact(beta, tau)
    b=f_approx(beta, tau, linear=TRUE)
    c=f_approx(beta, tau, linear=FALSE)
    x=c(x, max(abs(a-b)))
    y=c(y, max(abs(a-c)))
    z=c(z, nm(a-b))
    u=c(u, nm(a-c))
    }
  return(list(n=m:nn, g_abs=x, h_abs=y, g_nm=z, h_nm=u))
  }

test4 <- function(nn) {
  beta0=rnorm(nn)
  tau0=rnorm(nn)
  m=10
  nn=max(m, nn)
  x=numeric(0)
  y=x
  z=x
  u=x
  for (n in m:nn) {
    Cat(n)
    beta=beta0[1:n]
```

```
    tau=tau0[1:n]
    a=pinv(f_exact(beta,tau))
    b=pinv(f_approx(beta,tau,linear=TRUE))
    c=pinv(f_approx(beta,tau,linear=FALSE))
    x=c(x,max(abs(a-b)))
    y=c(y,max(abs(a-c)))
    z=c(z,nm(a-b))
    u=c(u,nm(a-c))
    }
  return(list(n=m:nn,g_abs=x,h_abs=y,g_nm=z,h_nm=u))
  }
```

References

[1] H. Strasser. Asymptotic expansions for conditional moments of Bernoulli trials. Technical report, Institute of Statistics and Mathematics, WU, 2011.

[2] H. Strasser. The covariance structure of conditional maximum likelihood estimates when the number of item parameters is large. Technical report, Institute of Statistics and Mathematics, WU, 2011.