JMASM27: An Algorithm for Implementing Gibbs Sampling for 2PNO IRT Models (Fortran)

Yanyan Sheng  
Southern Illinois University at Carbondale, ysheng@siu.edu

Todd C. Headrick  
Southern Illinois University at Carbondale, headrick@siu.edu

Follow this and additional works at: http://digitalcommons.wayne.edu/jmasm

Part of the Applied Statistics Commons, Social and Behavioral Sciences Commons, and the Statistical Theory Commons

Recommended Citation
DOI: 10.22237/jmasm/1177993920
Available at: http://digitalcommons.wayne.edu/jmasm/vol6/iss1/33

This Algorithms and Code is brought to you for free and open access by the Open Access Journals at DigitalCommons@WayneState. It has been accepted for inclusion in Journal of Modern Applied Statistical Methods by an authorized editor of DigitalCommons@WayneState.
JMASM27: An Algorithm for Implementing Gibbs Sampling for 2PNO IRT Models (Fortran)

Yanyan Sheng  Todd C. Headrick
Southern Illinois University-Carbondale

A Fortran 77 subroutine is provided for implementing the Gibbs sampling procedure to a normal ogive IRT model for binary item response data with the choice of uniform and normal prior distributions for item parameters. The subroutine requires the user to have access to the IMSL library. The source code is available at http://www.siu.edu/~epse1/sheng/Fortran/, along with a stand alone executable file.

Key words: IRT, two-parameter normal ogive model, MCMC, Gibbs sampling, Fortran.

Introduction

Item response theory (IRT) describes a probabilistic relationship between correct responses on a set of test items and a latent variable, where the influence of items and persons on the responses is modeled by distinct sets of parameters. Common IRT models include the two-parameter normal ogive (2PNO; Lawley, 1943, 1944; Lord, 1952, 1953a, 1953b) model such that the probability of person $i$ obtaining a correct response for item $j$, where $i = 1, ..., n$ and $j = 1, ..., k$, is defined as

$$P(y_{ij} = 1) = \Phi(\alpha_j \theta_i - \gamma_j)$$

$$= \int_{-\infty}^{\alpha_j \theta_i - \gamma_j} \frac{1}{\sqrt{2\pi}} e^{-t^2} dt, \quad (1)$$

where $\gamma_j$ and $\alpha_j$ denote item parameters and $\theta_i$ denotes the continuous person trait. In the model, items are assumed to vary in terms of location, $\gamma_j$, as well as slope, $\alpha_j$.

Simultaneous estimation of both item and person parameters results in statistical complexities in the estimation task of IRT models, which have made estimation procedures a primary focus of psychometric research over decades (e.g., Birnbaum, 1969; Bock & Aitkin, 1981; Molenaar, 1995). Recent attention has been focused on Markov chain Monte Carlo (MCMC; e.g., Chib & Greenberg, 1995) techniques, which have demonstrated to be useful for complex estimation problems in many areas of applied statistics. Albert (1992) was the first to apply an MCMC algorithm, known as Gibbs sampling (Casella & George, 1992; Gelfand & Smith, 1990; Geman & Geman, 1984), to the 2PNO model, where he adopted non-informative priors for item parameters.

As Albert’s (1992) focus was on investigating the applicability of Gibbs sampling to IRT, he did not specifically consider the situations where informative priors are adopted for item parameters. However, in some applications, they are more preferred than vague priors. For example, when comparing several candidate models, Bayes factors are commonly adopted in the Bayesian framework, but they are not defined with non-informative priors (Gelman et al., 2003). In this case, the program given by Albert (1992) does not provide a solution. Moreover, given that MCMC is computationally
demanding in drawing a sufficiently long chain to ensure convergence, a major problem in applied MCMC programs. Researchers have either used WinBUGS (e.g., Bazán, Branco & Bolfarine, 2006; DeMars, 2005) to implement MCMC for IRT models, or coded the sampler in S-Plus (e.g., Patz & Junker, 1999) or MATLAB (e.g., Albert, 1992). They noted that each execution consumed many hours, and hence was computationally expensive. This fact makes it impractical for users to utilize these programs for various applications of IRT. They further limit researchers in conducting Monte Carlo studies, or developing more complicated IRT models. It is then anticipated that Fortran will provide a better solution, as it is the fastest programming language for numerical computing (Brainerd, 2003).

In view of the above, the purpose of this article is to provide a Fortran subroutine that obtains the posterior estimates (and their associated standard errors) of item and person parameters in the 2PNO IRT model. The subroutine will have the option of specifying non-informative and informative priors for item parameters.

Methodology

The Gibbs Sampling Procedure

To implement Gibbs sampling to the 2PNO model defined in (1), a latent continuous random variable $Z_{ij}$ is introduced so that $Z_{ij} \sim N(\alpha_j \theta_i - \gamma_j, 1)$ (Albert, 1992; Tanner & Wong, 1987). With prior distributions assumed for $\theta_i$ and $\xi_j$, where $\xi_j = (\alpha_j, \gamma_j)'$, the joint posterior distribution of $(\theta, \xi)$ is hence

$$p(\theta, \xi | y) \propto f(y | Z)p(Z | \theta, \xi)p(\theta)p(\xi),$$

where $f(y | Z)$ is the likelihood function.

With a normal prior for $\theta_i$ and non-informative priors for $\alpha_j$ and $\gamma_j$ so that $\theta_i \sim N(\mu, \sigma^2)$, $\alpha_j > 0$ and $p(\gamma_j) \propto 1$, the full conditional distributions of $Z_{ij}, \theta_i$, and $\xi_j$ can be derived in closed forms as follows:

$$Z_{ij} | \cdot \sim \begin{cases} N_{(0,\infty)}(\alpha_j \theta_i - \gamma_j, 1), & \text{if } y_{ij} = 1 \\ N_{(-\infty,0)}(\alpha_j \theta_i - \gamma_j, 1), & \text{if } y_{ij} = 0 \end{cases} \quad (3)$$

$$\theta_i | \cdot \sim N\left(\frac{\sum_j (Z_{ij} + \gamma_j)\alpha_j + \mu}{\frac{1}{\sigma^2} + \sum_j \alpha_j^2}, \frac{1}{\frac{1}{\sigma^2} + \sum_j \alpha_j^2}\right) \quad (4)$$

$$\xi_j | \cdot \sim N((x'x)^{-1}x'Z_j, (x'x)^{-1}I(\alpha_j > 0)), \quad (5)$$

where $x = [\theta, -1]$. Alternatively, informative conjugate priors can be assumed for $\alpha_j$ and $\gamma_j$ so that $\alpha_j \sim N_{(0,\infty)}(\mu_\alpha, \sigma^2_\alpha)$, $\gamma_j \sim N(\mu_\gamma, \sigma^2_\gamma)$. In this case, the full conditional distribution of $\xi_j$ is derived as

$$\xi_j | \cdot \sim N((x'x + \Sigma_\xi^{-1})^{-1}(x'Z_j + \Sigma_\xi^{-1}\mu_\xi), (x'x + \Sigma_\xi^{-1})^{-1}I(\alpha_j > 0)) \quad (6)$$

where $\mu_\xi = (\mu_\alpha, \mu_\gamma)'$ and $\Sigma_\xi = \begin{pmatrix} \sigma^2_\alpha & 0 \\ 0 & \sigma^2_\gamma \end{pmatrix}$.

Hence, with starting values $\theta^{(0)}$ and $\xi^{(0)}$, observations $(Z^{(0)}, \theta^{(0)}, \xi^{(0)})$ can be simulated from the Gibbs sampler by iteratively drawing from their respective full conditional distributions specified in (3), (4) and (5) (or equations 3, 4, and 6). To go from $(Z^{(l-1)}, \theta^{(l-1)}, \xi^{(l-1)})$ to $(Z^{(0)}, \theta^{(l)}, \xi^{(l)})$, it takes three transition steps:

1. Draw $Z^{(l)} \sim p(Z | y, \theta^{(l-1)}, \xi^{(l-1)})$;
2. Draw $\theta^{(l)} \sim p(\theta | Z^{(l)}, \xi^{(l-1)})$;
3. Draw $\xi^{(l)} \sim p(\xi | Z^{(l)}, \theta^{(l)})$.

This iterative procedure produces a sequence of $(\theta^{(l)}, \xi^{(l)}), l = 0, \ldots, L$. To reduce the effect of the starting values, early iterations in the Markov chain are set as burn-ins to be discarded. Samples from the remaining iterations are then used to summarize the posterior density of item
parameters $\xi$ and ability parameters $\theta$. As with standard Monte Carlo, with large enough samples, the posterior means of $\xi$ and $\theta$ are considered as estimates of the true parameters. However, their standard deviations tend to underestimate the posterior standard deviations, as subsequent samples in Gibbs sampler are autocorrelated (e.g., Albert, 1992; Patz & Junker, 1999). One approach to calculating them is through batching (Ripley, 1987). That is, with a long chain of samples being separated into contiguous batches of equal length, the posterior mean and standard deviation for each parameter are then estimated to be the sample mean and standard deviation of these batch means respectively. Thus, the standard error of the estimate is a ratio of the standard deviation and the square root of the number of batches.

The Fortran Subroutine

The subroutine initially sets the starting values for the parameters, so that $\theta_i^{(0)} = 0$, $\alpha_i^{(0)} = 2$ and $\gamma_i^{(0)} = -\Phi^{-1}(\sum_j y_{ij} / n)\sqrt{5}$ (Albert, 1992). It then iteratively draws random samples for $Z$ and $\theta$ from their respective full conditional distributions specified in (3) and (4) with $\mu = 0$ and $\sigma^2 = 1$. Samples for $\xi_j$ are simulated either from (5), where uniform priors are assumed for $\xi_j$, or from (6), where normal priors are adopted with $\mu_\alpha = \mu_\gamma = 0$ and $\sigma^2_\alpha = \sigma^2_\gamma = 1$. The algorithm continues until all the $L$ samples are simulated. It then discards the early burn-in samples, and computes the posterior estimates and standard errors for the model parameters, $\theta$, $\alpha$ and $\gamma$, using batching.

For example, for a 2000-by-10 (i.e., $n = 2,000$ and $k = 10$) dichotomous (0-1) data matrix simulated using the item parameters shown in the first two columns of Table 1, the Gibbs sampler was implemented so that 10,000 samples were simulated with the first 5,000 taken to be burn-in. The remaining 5,000 samples were separated into 5 batches, each with 1,000 samples. Two sets of the posterior means for $\alpha$ and $\gamma$, as well as their standard errors, were obtained assuming the uniform or normal prior distributions described previously, and are displayed in the rest of the table. It is noted that the item parameters were estimated with enough accuracy and the two sets of posterior estimates differ only slightly from each other, signifying that the results are not sensitive to the choice of priors for $\xi$. For this example, each implementation took less than 13 minutes. Although 10,000 iterations are long enough for the Markov chain to reach the stationary distribution, one may easily increase the length of the chain to as long as 50,000, which takes about 60-90 minutes for each execution.

Conclusion

This Fortran subroutine leaves it to the user to choose between uniform and normal priors for the item parameters, $\alpha$ and $\gamma$. In addition, the user can change the source code so that the prior distribution for $\theta_i$ assumes different location, $\mu$ and scale, $\sigma^2$. Similarly, $\mu_\alpha$, $\sigma^2_\alpha$, and $\mu_\gamma$, $\sigma^2_\gamma$ can be modified to reflect different prior beliefs on the distributions for the item parameters. It is noted that convergence can be assessed by comparing the marginal posterior mean and standard deviation of each parameter computed for every 1,000 samples after the burn-ins. Similar values provide a rough indication of similar marginal posterior densities, which further indicates possible convergence of the Gibbs sampler (Gelfand, Hills, Racine-Poon & Smith, 1990; Hoijtink & Molenaar, 1997).
Table 1. Posterior estimates and their standard errors for $\alpha$ and $\gamma$ with uniform priors and normal priors.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\alpha$</th>
<th>$\gamma$</th>
<th>Posterior estimates</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>uniform priors</td>
</tr>
<tr>
<td></td>
<td>$\hat{\alpha}$</td>
<td>$\hat{\gamma}$</td>
<td>$\hat{\alpha}$</td>
</tr>
<tr>
<td>$\alpha$</td>
<td></td>
<td></td>
<td>(SE)</td>
</tr>
<tr>
<td>0.0966</td>
<td>0.1147</td>
<td>-0.8258</td>
<td>0.1120</td>
</tr>
<tr>
<td></td>
<td>(.0009)</td>
<td>(.0010)</td>
<td>(.0012)</td>
</tr>
<tr>
<td>0.0971</td>
<td>0.1291</td>
<td>-0.5441</td>
<td>0.1285</td>
</tr>
<tr>
<td></td>
<td>(.0003)</td>
<td>(.0006)</td>
<td>(.0022)</td>
</tr>
<tr>
<td>0.4589</td>
<td>0.4412</td>
<td>0.9289</td>
<td>0.4416</td>
</tr>
<tr>
<td></td>
<td>(.0031)</td>
<td>(.0021)</td>
<td>(.0031)</td>
</tr>
<tr>
<td>0.9532</td>
<td>1.1335</td>
<td>0.8906</td>
<td>1.1124</td>
</tr>
<tr>
<td></td>
<td>(.0088)</td>
<td>(.0032)</td>
<td>(.0074)</td>
</tr>
<tr>
<td>0.0771</td>
<td>0.0517</td>
<td>-0.8063</td>
<td>0.0510</td>
</tr>
<tr>
<td></td>
<td>(.0005)</td>
<td>(.0005)</td>
<td>(.0007)</td>
</tr>
<tr>
<td>0.4891</td>
<td>0.4761</td>
<td>-0.6186</td>
<td>0.4726</td>
</tr>
<tr>
<td></td>
<td>(.0023)</td>
<td>(.0006)</td>
<td>(.0023)</td>
</tr>
<tr>
<td>0.8599</td>
<td>0.7960</td>
<td>0.3484</td>
<td>0.8015</td>
</tr>
<tr>
<td></td>
<td>(.0028)</td>
<td>(.0013)</td>
<td>(.0036)</td>
</tr>
<tr>
<td>0.9427</td>
<td>0.9230</td>
<td>-0.8661</td>
<td>0.9189</td>
</tr>
<tr>
<td></td>
<td>(.0060)</td>
<td>(.0017)</td>
<td>(.0083)</td>
</tr>
<tr>
<td>0.2727</td>
<td>0.3981</td>
<td>-0.9497</td>
<td>0.3948</td>
</tr>
<tr>
<td></td>
<td>(.0027)</td>
<td>(.0018)</td>
<td>(.0028)</td>
</tr>
<tr>
<td>0.6532</td>
<td>0.6562</td>
<td>-0.3785</td>
<td>0.6544</td>
</tr>
<tr>
<td></td>
<td>(.0016)</td>
<td>(.0008)</td>
<td>(.0015)</td>
</tr>
</tbody>
</table>

References


---

**Appendix**

```fortran
SUBROUTINE GSU2(Y, N, K, L, BURNIN, BN, UNIF, ITEM, PERSON)
C*************************************************************************
C Y is the n-by-k binary item response data
C N is the number of subjects
C K is the test length (number of items)
C L is the number of iterations using Gibbs sampling
C BURNIN is the first number of iterations that are to be discarded
C BN is the number of batches
C UNIF is a 0-1 indicator with 0 specifying normal priors for item
C parameters and 1 specifying uniform priors for them
C ITEM is a k-by-4 matrix of posterior estimates and standard errors
C for item parameters
C PERSON is a n-by-2 matrix of posterior estimates and standard errors
C for person abilities
C*************************************************************************
```
INTEGER    L, COUNT, IRANK, BURNIN, UNIF, INDX(2), BN, 
& BSIZE, Y(N, K)
REAL A(K), G(K), TH(N), LP, MU, VAR, AV(L,K), GV(L,K), THV(N,L), 
& PHAT(K), U, Z(N,K), V, MN, MSUM, PMVAR, FMEAN, TT, T X(N,2), 
& XX(2,2), IX(2,2), ZV(N,1), XZ(2,1), AMAT(2,2), BZ(2,1), AMU, 
& GMU, AVAR, GVAR, AGMU(2,1), AGVAR(2,2), SIGMA(2,2), BETA(1,2), 
& BI(1,2), ITEM(K,4), PERSON(N,2), SUM1, SUM2, SUM3, 
& M1, M2, M3, TOT1, TOT2, TOT3, SS1, SS2, SS3
DOUBLE PRECISION    BB, TMP
C*************************************************************************
C Connect to external libraries for normal (RNNOR) and uniform (RNUN) 
C random number generator, inverse (ANORIN, DNORIN) and CDF (ANORDF, 
C DNORDF) for the standard normal distribution, and Cholesky 
C factorization (CHFAC) routines
C*************************************************************************
EXTERNAL  RNNOR, RNSET, RNUN, ANORDF, ANORIN, CHFAC, DNORDF, DNORIN
C*************************************************************************
C Set initial values for item parameters a, g, and person ability theta so 
C that a = 2, g = \frac{-\Phi^{-1}(\sum_i y_{ij}/n)\sqrt{5}}{f} for all k items, and theta = 0 for all n 
C persons.
C*************************************************************************
    PHAT = SUM(Y, 1)
    DO 10 I = 1, K
       A(I) = 2.0
       G(I) = -ANORIN(PHAT(I)/N)*SQRT(5.0)
10 CONTINUE
    DO 20 I = 1, N
       TH(I) = 0.0
20 CONTINUE
C*************************************************************************
C MU and VAR are the mean and the variance for the prior distribution of 
C theta. 
C*************************************************************************
    MU = 0.0
    VAR = 1.0
C*************************************************************************
C Start iteration 
C*************************************************************************
    COUNT = 0
    DO 30 IT = 1, L
       COUNT = COUNT + 1
30 CONTINUE
C*************************************************************************
C Update samples for Z from its normal posterior distributions 
C*************************************************************************
    DO 40 I = 1, N
       DO 40 J = 1, K
          LP  = TH(I) * A(J) - G(J)
          BB = ANORDF((0.0 - LP))
          CALL RNUN (1, U)
          TMP = BB*(1 - Y(I, J)) + (1 - BB)*Y(I, J)) * U + BB*Y(I, J)
          Z(I, J) = DNORIN(TMP) + LP
40 CONTINUE
C***************************************************************************
C Update samples for theta from their normal posterior distributions
C***************************************************************************
V = 1/SUM(A*A)
PVAR = 1/(1/V + 1/VAR)
DO 50 I = 1, N
    MSUM = 0.0
    DO 60 J = 1, K
        MSUM = MSUM+A(J)*(Z(I, J) + G(J))
    60 CONTINUE
MN = MSUM*V
PMEAN = (MN/V + MU/VAR)*PVAR
CALL RNNOR(1,TT)
TH(I) = TT*SQRT(PVAR) + PMEAN
THV(I, COUNT) = TH(I)
50 CONTINUE
C***************************************************************************
C Update samples for item parameters, a and g from their multivariate
C normal posterior distributions
C***************************************************************************
DO 70 J = 1, 1
    DO 70 I = 1, N
        X(I, J) = TH(I)
70 CONTINUE
DO 80 J = 2, 2
    DO 80 I = 1, N
        X(I, J) = -1
80 CONTINUE
IF (UNIF = 0) THEN
C***************************************************************************
C Specify the prior means (AMU, GMU) and variances (AVAR, GVAR)
C for a and g.
C***************************************************************************
AMU = 0.0
GMU = 0.0
AVAR = 1.0
GVAR = 1.0
C***************************************************************************
C Put the means and variances in vector and matrix format
C***************************************************************************
AGMU(1, 1) = AMU
AGMU(2, 1) = GMU
AGVAR(1, 1) = AVAR
AGVAR(2, 2) = GVAR
C***************************************************************************
C Call the matrix inversion routine.
C Invert matrix AGVAR with the inverse stored in SIGMA
C*************************************************************************
CALL MIGS(AGVAR, 2, SIGMA, INDX)
XX = MATMUL(TRANSPOSE(X), X) + SIGMA
ELSE IF (UNIF = = 1) THEN
XX = MATMUL(TRANSPOSE(X), X)
END IF
C*************************************************************************
C Call the matrix inversion routine.
C Invert matrix XX with the inverse stored in IX
C*************************************************************************
CALL MIGS(XX, 2, IX, INDX)
C*************************************************************************
C Call the Cholesky factorization routine. Compute the Cholesky
C factorization of the symmetric definite matrix IX and store the
C result in AMAT
C*************************************************************************
CALL CHFAC (2, IX, 2, 0.00001, IRANK, AMAT, 2)
DO 90 J = 1, K
DO 100 I = 1, N
ZV(I, 1)=Z(I, J)
100       CONTINUE
IF (UNIF = = 0) THEN
XZ = MATMUL(SIGMA, AGMU)+MATMUL(TRANSPOSE(X), ZV)
ELSE IF (UNIF = = 1) THEN
XZ = MATMUL(TRANSPOSE(X), ZV)
END IF
BZ = MATMUL(IX, XZ)
A(J) = 0
DO WHILE (A(J).LE.0)
CALL RNNOR (2, BI)
BETA = MATMUL(BI, AMAT)+TRANSPOSE(BZ);
A(J) = BETA(1, 1)
G(J) = BETA(1, 2)
END DO
AV(COUNT, J) = A(J)
GV(COUNT, J) = G(J)
90  CONTINUE
30 CONTINUE
C*************************************************************************
C Calculate the posterior means and SEs for a, g and theta and store them
C in ITEM and PERSON
C*************************************************************************
BSIZE=(L-BURNIN)/BN
DO 110 J = 1, K
COUNT = BURNIN
TOT1 = 0.0
TOT2 = 0.0
SS1 = 0.0
SS2 = 0.0
DO 120 M = 1, BN
SUM1 = 0.0
SUM2 = 0.0
DO 130 I = 1, BSIZE
COUNT = COUNT + 1
SUM1 = SUM1 + AV(COUNT, J)
SUM2 = SUM2 + GV(COUNT, J)
130       CONTINUE
CONTINUE
M1 = SUM1/BSIZE
M2 = SUM2/BSIZE
TOT1 = TOT1 + M1
TOT2 = TOT2 + M2
SS1 = SS1 + M1*M1
SS2 = SS2 + M2*M2

ITEM(J, 1) = TOT1/BN
ITEM(J, 2) = SQRT((SS1 - (TOT1*TOT1/BN))/(BN -1))/SQRT(FLOAT(BN))
ITEM(J, 3) = TOT2/BN
ITEM(J, 4) = SQRT((SS2 - (TOT2*TOT2/BN))/(BN -1))/SQRT(FLOAT(BN))

CONTINUE

DO 140 J = 1,N
  COUNT = BURNIN
  TOT3 = 0.0
  SS3 = 0.0
  DO 150 M = 1, BN
    SUM3 = 0.0
    DO 160 I = 1, BSIZE
      COUNT = COUNT + 1
      SUM3 = SUM3 + THV(J, COUNT)
    160  CONTINUE
    M3 = SUM3/BSIZE
    TOT3 = TOT3 + M3
    SS3 = SS3 + M3*M3
  150  CONTINUE
  PERSON(J, 1) = TOT3/BN
  PERSON(J, 2) = SQRT((SS3 - (TOT3*TOT3/BN))/(BN -1))/SQRT(FLOAT(BN))
  CONTINUE

RETURN
END