

THE EFFECT OF ERROR IN ITEM PARAMETER ESTIMATES ON LINKING AND
EQUATING WITH THE IRT TEST CHARACTERISTIC CURVE METHOD

by

Gary Scott Kaskowitz

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
1998

CJ
MD
Dept. of Measurement, Statistics
and Evaluation

Advisory Committee:

Professor Rafael J. De Ayala, Chair
Professor Frank B. Baker
Professor C. M. Dayton
Professor Robert Lissitz
Professor Stephen Porges

Maryland

LD

3231

M70d

Kaskowitz,

G.S.

UMI Number: 9836419

Copyright 1998 by
Kaskowitz, Gary Scott

All rights reserved.

UMI Microform 9836419
Copyright 1998, by UMI Company. All rights reserved.

This microform edition is protected against unauthorized
copying under Title 17, United States Code.

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

ACKNOWLEDGEMENTS

The author gratefully acknowledges Frank Baker of the University of Wisconsin for the use of the EQUATE program.

TABLE OF CONTENTS

List of Tables	vi
List of Figures	vii
Chapter I - Introduction	1
Chapter II - Literature Review	5
Background	5
Principles of Equating	8
Issues for Equating	14
Common Issues for Equating	14
Equating Issues Associated with IRT	22
Equating Processes	24
Traditional Techniques	24
IRT Methods	29
Measuring the Accuracy of Equating Results	38
Summary	41
Chapter III - Methodology	43
Statement of Problem	43
Purpose and Overview of study	43
Determination of standard error for item parameter estimates	44
Selection of item parameter estimates	47
Estimation of equating coefficients	47
Variables in the study	48
Independent variables	48
Dependent variables	49
Procedures for study	53
Linking study	53
Equating study	57
Summary	60
Chapter IV - Results	61
Nomenclature	61
Parameter and Data Generation	61
Non-related error conditions	61
Related error conditions	62
All conditions	65
Linking Study Results	66
Recovery of a and b parameters	66

Non-related error conditions	66
Related error conditions	67
RMSE and Bias in a and b	72
Non-related error conditions	72
Related error conditions	73
F_{\max} , F_{converge} and IR	77
Non-related conditions	77
Related error conditions	78
All conditions	79
Equating study results	83
Chapter V - Discussion	87
Linking Study	87
Recovery of a and b parameters	87
Non-related error conditions	87
Related-error conditions	88
Bias and RMSE of a and b estimation	89
Non-related error conditions	89
Related-error conditions	90
F_{\max} , F_{converge} and IR	90
Equating Study	91
General Discussion	92
Implications for future research	93
Appendix A	
References for design decisions	95
Appendix B	
Item parameter distributions	96
Appendix C	
Mean values of alpha as a function of # of common items and error level	102
Appendix D	
Mean values of beta as a function of # of common items and error level	104
Appendix E	
RMSE and Bias of alpha as a function of # of common items and error level	106

Appendix F	
RMSE and Bias of beta as a function of # of common items and error level	110
Appendix G	
Mean Improvement Ratios (IR) as a function of # of common items and error level	114
Appendix H	
Mean F_{\max} values as a function of # of common items and error level	116
Appendix I	
Mean F_{converge} values as a function of # of common items and error level	118
Appendix J	
Mean Theta RMSE values as a function of # of common items and error level	120
Appendix K	
Mean True Score RMSE values as a function of # of common items and error level	122
Appendix L	
Source code listing	124
References	183
Footnotes	193

LIST OF TABLES

Table 1 - 2PL Standard Errors (SE) - Non-related error	45
Table 2 - 3PL Standard Errors (SE) - Non-related error	45
Table 3 - Standard Errors for b when SE related to b value	46
Table 4 - Standard Errors for α when SE related to b values	46
Table 5 - Quadrature values and weights used in EAP estimation	59
Table 6 - Descriptive Statistics for Item Parameters	63
Table 7 - Means and Standard Deviations for α and β (non-related error)	69
Table 8 - Means and Standard Deviations for α and β (related error)	70
Table 9 - Bias and RMSE for α and β Estimates (non-related error)	75
Table 10 - Bias and RMSE for α and β Estimates (related error)	76
Table 11 - Means and Standard Deviations for F_{\max} and F_{converge} (non-related error)	80
Table 12 - Means and Standard Deviations for F_{\max} and F_{converge} (related error)	81
Table 13 - Improvement Ratios Means and Deviations	82
Table 14 - RMSE of θ and True Scores non-related error conditions	85
Table 15 - RMSE of θ and True Scores related error conditions	86

LIST OF FIGURES

Figure 1 - Non-related a parameters vs. non-related b parameters	62
Figure 2 - Non-related c parameters vs. non-related b parameters	62
Figure 3 - Related a parameters vs. Related b parameters	63
Figure 4 - Related c parameter vs. Related b parameter	64
Figure 5 - Example of H3Y05 Case	64
Figure 6 - Example of L3Y05 Case	65
Figure 7 - Distribution of ability parameters used in study	65
Figure 8 - Mean alphas for non-related 2PL conditions	71
Figure 9 - Mean alphas for related 3PL conditions	71

Chapter I - Introduction

For practical or policy reasons, it is often necessary to administer different forms of a test to different groups. For example, in a large scale testing program such as the SATs, different forms of the test are administered on different testing dates. Yet, even though different forms have been used to obtain the scores of the examinees, it is necessary to be able to report all scores interchangeably. Even when two forms are designed to be psychometrically equivalent, it is possible that form-to-form differences can create one form which is more difficult than another. For example, if a person takes the more difficult form, then that person should not have a reported score that is less than that he or she would have obtained by taking the easier form. Test equating attempts to effectively compare scores across forms by statistically adjusting scores on one form to make them comparable to scores on another.

This paper defines the different types of equating which are conducted in practice as well as the necessary assumptions and conditions for equating. The literature review discusses the problems and issues associated with equating and the general principles behind equating, such as how the data were collected for the forms which are being equated.

Test equating can be accomplished through several techniques, each based upon different assumptions and conditions. The various methods of test equating are discussed in this paper. This paper discusses the traditional approaches of equating which involve the use of empirical analyses of observed scores. Techniques such as mean, linear, and equipercentile equating are described. These traditional methods attempt to analyze the

observed scores on the forms which are being equated and adjust various statistical properties of the distributions such as the mean.

A more theoretical approach towards equating, and the focus of this study, is that of Item Response Theory (IRT). IRT equating assumes that the forms which are being equated follow an IRT framework. Typically, a mathematical model is proposed which relates the responses on an item to an underlying continuous ability variable. IRT equating is based upon a set of assumptions and conditions which are discussed in the literature review.

There are several IRT equating techniques. These techniques include the regression, mean and sigma, and test characteristic curve (TCC) methods of equating. All of these techniques exploit the fact that in IRT parameter estimates are invariant when there is model-data fit. The different IRT equating techniques use the information available in the item parameter estimates to different degrees in order to compute the equating coefficients. Of the various IRT equating techniques the TCC technique has received a great deal of attention because it uses more of the available information than the other methods. The TCC technique for equating is the approach which was investigated in this study.

Many studies have been conducted to determine the effectiveness of particular equating techniques. In the field of IRT equating it is important to know which factors can influence the estimation of equating coefficients. The equating coefficients are calculated using estimated item parameters. Yet a fundamental assumption made when estimating equating coefficients is that the information used to compute them has no

estimation error (i.e., the item parameters are known). In reality, the error associated with the estimation of item parameters would be expected to be propagated to the estimation of equating coefficients.

The focus of this research was to look at the effect of item parameter estimation error on the estimation of the equating coefficients for the TCC method. This study also looked at the robustness and accuracy of equated scores based upon these equating coefficient estimates. It is possible that the TCC method will be robust to a large amount of error in the estimation of the item parameters.

A major consideration in equating is the determination of the accuracy of the equating. Several techniques have been described to help gauge this accuracy such as root mean square error (RMSE) and standardized mean square difference. This study introduces a technique for analyzing the accuracy of the obtained equating coefficient estimates based on the TCC method of equating. This technique calculates the maximum loss function between the two TCCs that are being equated (i.e., how far apart the curves are initially). This value will represent the maximum possible improvement in the equating. After equating the minimized value of the loss function is related to the maximum possible improvement. This approach allows for direct comparisons across different pairs of forms by establishing a common ratio applicable to all TCC equatings.

In addition to examining the accuracy of the equating coefficients, this study assessed the quality of equating by estimating the true scores using the equated estimates. These scores were compared against known values to determine the accuracy of the equating.

This study contributes to the literature by examining the robustness of the TCC method to estimation error. It was expected that TCC would be found to be robust.

Chapter II - Literature Review

Background

Oftentimes it is necessary to compare scores across multiple versions of a test. For example, large scale testing programs typically will administer different forms of a test for security reasons. When different forms of a test are administered to different groups, yet the scores are compared interchangeably for decision-making, it becomes imperative that the scores from all versions of the test are comparable. This process of making scores comparable across multiple versions of a test is known as equating.

Equating has been defined differently by different authors. Several terms and definitions have arisen within this field. Yet the general consensus for defining equating would be the process of statistically adjusting the scores on different test forms so that the scores can be used interchangeably (Dorans, 1990; Kolen & Brennan, 1995). Even though the process of applying statistical adjustments to the data is generally agreed upon, there remain several different definitions of just what equating is (that is, how does one know when two forms are sufficiently “equated”). Lord (1980) first developed four conditions for equating. These have subsequently been utilized and expanded upon by several authors (Harris & Crouse, 1993; Kolen & Brennan, 1995; Lord, 1980). The four key properties for equating established by Lord can be described as:

1. Symmetry Property requires that the function used to transform a score on Form X to Form Y be the inverse of the function used to transform a score on Form Y to Form X;
2. Same Specifications Property implies that test forms must be built to the same

content and statistical specifications in order to be equated;

3. Equity Properties, which are either strong as defined by Lord (1980), or weak, as defined by Morris (1982), but both generally positing that a person's true score should be consistent regardless of the form of the test which was taken (in fact, Lord further defined equity as existing if two forms (X and Y) are equitable, then it must be a matter of indifference to each person whether they are administered Form X or Form Y); and
4. Group Invariance Property which holds that the equating relationship between forms of a test is the same regardless of the group of examinees which is used to conduct the equating.

These properties focus on the different aspects of equating such as looking at individual vs. group scores or observed vs. unobserved scores on a test form and can be affected by the choice of a particular equating theory or method (Kolen & Brennan, 1995). The literature suggests several different types and methods of equating.

The two main types of equating are horizontal and vertical. Horizontal equating is used when the ability distributions of the groups being equated are similar. In this case, parallel forms of a test are designed to be as similar as possible on their psychometric properties (i.e., the forms are all supposed to be measuring at the same difficulty level) and ideally the only difference in an individual examinee's score would be due to the person's particular ability and not the form of the test (Baker, 1984; Kolen & Brennan, 1995; Skaggs & Lissitz, 1986, 1988). This is the most typical case when the different forms are administered to different samples from the same population. The premise is that the

groups are expected to be similar to one another and to be assured that any particular group is not penalized by taking a more difficult form than another group.

Vertical equating is distinguished from horizontal equating in that the groups of examinees are expected to differ in their ability levels and the forms differ in their difficulty levels. This is commonly used in such applications as equating test scores across grade levels in school (Baker, 1984; Skaggs & Lissitz, 1988). Kolen and Brennan (1995) preferred to call this process vertical “scaling to achieve comparability” or linking. Kolen and Brennan pointed out that these forms which are vertically equated are designed to be different whereas equating is used to adjust scores for test forms which are built to be as similar as possible in content and statistical properties.

Unfortunately, the terminology and definitions of equating become more dissimilar with the introduction of IRT approaches to equating. When IRT is used for equating, “scaling” takes on a new meaning. In IRT, scaling and linking are generally thought of as putting an ability estimate ($\hat{\theta}$) onto a common metric (Baker, 1990; Cook & Eignor, 1991; Hambleton & Swaminathan, 1985). IRT posits that the metric obtained from item parameter estimation procedures is unique only up to a linear transformation (Kim & Cohen, 1995). Therefore, if the item parameter estimates are correct, and IRT holds, then the scaling is accomplished once the two forms are scaled to the same metric. This process is often referred to as linking. In practical situations, equating can then be accomplished by obtaining actual equivalent ability scores on the forms which are being equated (Hambleton & Swaminathan, 1985; Kolen & Brennan 1995; Vale, 1986).

As can be seen, the choice of a definition of equating often comes from the

theoretical position which is used to describe the testing situation. There are two fundamental approaches towards equating with the main distinction between them being the empirical analysis of observed scores (traditional) or theory-based equating of true scores (IRT) (Cook & Eignor, 1991; Skaggs & Lissitz, 1986, 1988). As Cook and Eignor pointed out, traditional and IRT test equating procedures were both developed to provide comparable scores on multiple forms of the same test in order to adjust for slight differences in form-to-form difficulty. Both traditional and IRT approaches require that the forms are not vastly different in content, difficulty, or reliability (Cook & Eignor, 1991).

Principles of Equating

Among the traditional methods for equating, Kolen and Brennan (1995) described three techniques: mean equating; linear equating and equipercentile equating. The correct application of mean equating requires the assumption that the two forms are consistently different in difficulty along the entire score scale. Then mean equating merely becomes the addition of a constant to all scores on one form to find the equated scores on the second form (Kolen & Brennan, 1995). However, the assumption that the differences in scores is constant across all points of the scale might appear overly restrictive and many researchers favor linear or equipercentile equating methods (Brennan & Kolen, 1987; Budescu, 1985; Dorans, 1990; Gafni & Melamed, 1990; Huynh, & Ferrara, 1994; Kolen & Whitney, 1982; Livingston, Dorans, & Wright, 1990; Schmitt, Cook, Dorans, & Eignor, 1990).

Kolen and Brennan (1995) described linear equating as an extension of mean

equating, where the differences between the forms is not assumed to be constant. Linear equating looks at scores distances from their means in terms of standard deviation units instead of raw scores. This method of equating allows the scale units as well as the means to differ from one form to the next. Similar to z-scores, linear equating is accomplished by setting the z-scores of the two forms equal to one another (Woodruff, 1986). This method of equating tends to provide more flexibility than mean equating.

When an even less constrained form of equating is desired, researchers often use equipercentile equating. Whereas mean and linear equating assume that the shapes of the score distributions for the forms being equated are the same, equipercentile equating makes no such assumption. For example, Form X might be more difficult than Form Y for high scores but less difficult for low scores. Kolen and Brennan (1995) and Cook and Petersen (1987) generally described this method as taking a given Form X score and finding the percentage of examinees with scores at or below that score. The Form Y score which has the same percentage of examinees at or below it as the Form X score is considered the equated score. Because equipercentile equating is a nonlinear equating technique, it is even more flexible and general than linear equating.

These traditional approaches to equating are empirically based, where the observations of scores on the multiple forms are used to find the relationship between them (either linear or curvilinear). In contrast, a more theoretical model of the scores and data is assumed with IRT equating. Whereas classical test models are based on responses at the level of test scores, IRT models examinee responses at the item level (Kolen & Brennan, 1995). IRT based methods of equating are an attempt to model an examinee's

performance on an item as a function of both characteristics of the item and the examinee's ability. It is assumed that the exam is measuring a latent trait. A mathematical function models the relationship between an examinee's proficiency and the probability that the examinee will answer the item correctly. In order to use IRT equating, several assumptions must be made.

A key assumption to all equating techniques is that the equating transformation be population independent so that the equating transformation is consistent regardless of the group(s) which is(are) used to compute the transformation (Cook & Petersen, 1987). If this is not the case, then the transformation may not be of much use beyond the groups which comprised the original data for the computations. If the transformation is not independent from the sample(s), it could be because the forms are not measures of the same construct (Dorans, 1990).

Unidimensional IRT equating assumes that the underlying trait being measured by the form is unidimensional (i.e., the form is measuring one construct) and that there is a single latent variable (θ) which represents the ability of the examinee on the trait being measured (Hambleton & Swaminathan, 1985).¹ Furthermore, IRT assumes that examinees responses to the items are statistically independent when conditioned on the examinee's ability. This assumption, known as local independence further implies that there are no relationships among items on a form other than those which are attributable to latent ability. For a given item it is assumed that the probability of correctly answering an item as a function of ability may be modeled by an IRT model. In general, the unidimensional models may be represented as:

$$P_i(\theta_j) = F[Da_i(\theta_j - b_i)] \quad (\text{II.1})$$

where:

F is the logistic function;

a_i is the item discrimination parameter;

b_i is the item difficulty parameter;

D = 1.702 (a scaling factor)

As presented, (II.1) is the two parameter logistic model (2PL). If all items are assumed to have the same value of a_i then one obtains the one parameter logistic (1PL) model. A special case of the 1PL model is the Rasch model where a_i is fixed at unity across all items. For the three parameter logistic model (3PL) equation (II.1) is modified to be:

$$P_i(\theta_j) = c_i + (1 - c_i)F[Da_i(\theta_j - b_i)] \quad (\text{II.2})$$

where

c_i is the pseudo-chance level parameter, or item-guessing factor.

Graphically, this relationship between $P_i(\theta_j)$ and θ_j may be represented by an item characteristic curve (ICC). When the probabilities for each item are computed, an expected number right score (i.e., a true score) for examinee j can be computed as:

$$\xi_j = \sum P_i(\theta_j) \quad (\text{II.3})$$

where $P_i(\theta_j)$ is the probability of a correct response on item i by examinee j according to an IRT model (Hambleton & Swaminathan, 1985). If $\hat{\theta}_j$ is used in lieu of θ_j , then $\hat{\xi}_j$ is the estimated true score. If these assumptions and relationships hold, then an IRT equating

method may be applied.

If one has model-data fit, θ will be independent of the form used to measure it, and any well-designed test should yield an estimate of θ . In essence, if the item parameters are accurately estimated $\hat{\theta}$ will not be affected by the subset of items used for a given form (Hambleton & Swaminathan, 1985). IRT equating therefore suggests that multiple forms of a test are merely separate subsets of items used to measure θ , and equating is merely the process of scaling the parameter estimates. Therefore, estimates of the examinee's ability from any of the forms should be equivalent once the parameters have been placed on the same scale (Cook & Eignor, 1991; Dorans, 1990). As Yamamoto and Mazzeo (1992) mentioned, one of the strengths of IRT models is that if the assumptions hold and the estimates of the item parameters are available for the different test forms, all results can be reported directly in terms of the ability scale. This property removes the need to compare number correct scores for the various forms. If desired, the test scores can be represented in terms of estimated true scores through an additional step (Hambleton & Swaminathan, 1985).

There does not appear to be strong agreement on which type of equating is best. The general consensus would be that the equating method depends upon the data collection method and type of equating. Skaggs and Lissitz (1986) discussed several generalizations for both horizontal and vertical equating conditions. Among their findings for horizontal equating, they recommended that if the data are reliable, the samples are nearly equal in ability and the forms are nearly equal in difficulty then all equating methods should provide adequate results. However, if the samples of examinees were not

randomly selected, then IRT equating methods should be used. Furthermore, they found that for vertical equating, both the Rasch and linear models produced inadequate results and that if vertical equating must be done, the best approach would be an equipercentile or an IRT 3PL model that did not depend on concurrent estimation of the item parameters (where the item parameters for the two forms are put on a common scale by estimating the parameters in one calibration run) (Skaggs & Lissitz, 1986).

Compared to the traditional methods of equating, Hambleton and Swaminathan (1985) suggested that IRT methods for equating are 1) linear; 2) group independent and 3) not affected by difficulty levels of the form, making them appropriate for vertical equating. When compared to the traditional types of equating, Cook and Eignor (1991) found several practical advantages to using IRT equating. These included: 1) IRT equating provided better equating than did the traditional methods at the upper ends of score scales; 2) IRT equating allowed greater flexibility in choosing previous forms of a test used for equating; 3) reequating (where a previously equated form is found to contain a possible error in its scoring and needs to be equated again) is easier if it is decided to not score an item after the form is administered; and 4) IRT equating allows the researcher to preequate at the item-level and to derive the relationship between test forms before they are administered operationally.

However, other researchers (e.g., Cook & Eignor, 1991) have pointed out that IRT gains its benefits by making strong statistical assumptions which may not hold in practice. Thus, the debate over which equating technique to use continues.

Issues for Equating

Common Issues for Equating

Regardless of the equating method chosen, there are common principles which must be considered. Harris and Crouse (1993) specified several criteria to help determine whether equating is even appropriate or necessary. They discussed the criteria of 1) weak equity (where only the means of the conditional distributions on each test after equating must be equal), 2) indices (summaries used to compare two sets of conversions), 3) standard errors (to estimate amount of equating error from sampling), 4) generated data (if data were simulated for a study), 5) equating a form to itself (if a form is equated to itself directly or through intervening forms), 6) large samples (used as an estimate of the population conversion), 7) consistency (where equating results are compared across the method), 8) stability (where the equating results are replicated), and 9) practical/heuristics issues. Harris and Crouse reviewed several studies which addressed one or more of these criteria and concluded that the specification of a criterion is critical to a study of equating, for without appropriate measures of accuracy, a thorough evaluation of the equating results is not possible. Harris and Crouse suggested that because equating results often appear to be situation specific studies should be replicated with results compared across studies.

Brennan and Kolen (1987b) discussed equating considerations common to all forms of equating. They mentioned that the researcher must be concerned about identifying, quantifying and eliminating various sources of error in equating such as model

misspecification, rounding of scores, or inconsistencies between idealized equating designs and the actual equating practice. Brennan and Kolen also discussed the consideration of content specifications and equating, equating in the context of cut scores, reequating and the effects of a security breach on equating. The authors made the argument that these are important issues that must be considered and are independent of the equating technique which is chosen.

Another consideration when conducting equating is the data collection design. There are three major types of data collection design (Hambleton & Swaminathan, 1985; Kolen & Brennan, 1995). These include:

1. Single-group design. This design gives the two different forms to the same group of examinees. This design has the benefit of having the same examinees take the two forms, so that the sample is common across both forms. However, possible disadvantages are that examinees might become fatigued or that the order of administration might impact the results on the form.
2. Equivalent-group Design. In this design, the two forms are given to equivalent, but not identical, groups of examinees. A practical feature of this design is that each examinee only takes one form, reducing the testing time relative to the single-group design and reducing possible fatigue and order effects. A drawback to this design is that the groups are thought to be equivalent samples from the same population, but this may not be the case.
3. Anchor-items Design. This particular design is sometimes called the common-item

nonequivalent populations design (Jarjoura & Kolen, 1985; Kolen & Brennan, 1995). This design calls for the administration of the two forms to two different groups of people with each form containing a core set of common items. These items can either be “internal” (counting towards the score of the examinee) or “external” (as a separate form which is not used towards the examinee’s score). It is necessary that the common items be representative of the total forms in terms of content and statistical properties. The anchor-group design has the benefit of using two groups of examinees while having some specific items in common, so that direct relationships can be determined. However, because the only link between the two groups is the common items, the content and statistical properties of the common items are critical if the groups differ in ability (Kolen & Brennan, 1995).

Various modifications can be made to these designs. However, it can be seen that in order to accomplish the equating of two forms, it is necessary that there be something in common across the forms to be equated. Each of these basic designs attempts to do that by either using the same (or similar) people or the common items across the forms of a test.

The common items design has become a very popular approach towards equating (Dorans, 1990; Eignor, Stocking, & Cook, 1990; Klein & Jarjoura, 1985; Kolen & Harris, 1990; Lawrence & Dorans, 1990; Livingston et al., 1990). This appears to be for the practical reason of allowing the two equating groups to be tested at different times with different forms.

As discussed above, there are two common methods of conducting a common-anchor item test, the internal anchor and the external anchor. Baker (1984) discussed the use of the internal and external anchors for vertical equating with an IRT approach. Using the external anchor approach, Baker found that the agreement of the observed and theoretical mean values of the discrimination parameter estimates was good while the agreement for the difficulty parameter estimates was poor. Using the internal anchor, Baker found that the agreement was not particularly good for either of the parameters.

Other issues involving the common items design include the number of items which should be in common between the two forms and how well these items represent the full form. Budescu (1985) argued that the key to a successful common-item test is the correlation between the common items and the rest of the operational form. As the correlation decreases, the usefulness of the common items become less meaningful, while when the correlation increases, the quality and precision of the estimated parameters for the combined group also increases. Budescu made a general argument that the longer the anchor, the higher the correlation it will have with the operational form. Similarly, Klein and Jarjoura (1985) compared the accuracy of content-representative anchors versus nonrepresentative but substantially longer anchors. They compared a 60-item representative common-item test to a 105 and 101-item nonrepresentative test. The authors concluded that when nonrandom groups in a common-item equating design perform differentially, it is very important that the common items directly represent the content area of the full operational form. They noted that a failure to equate on the basis of the content-representative items could lead to substantial error.

While most authors agree that the common items need to be representative of the remainder of the form(s) there does not appear to be a clear consensus on how many common items are necessary for the common-item test. Wingersky and Lord (1984) studied the use of 5, 15 and 25 common items for an anchor test. They reported that 25 items gave the best performance, but that the use of 15 common items was adequate for linking. Wingersky and Lord further reported that linking could be achieved with as few as five common items when the item parameters for the two forms were concurrently estimated. Baker (1996) and Kolen (1990) asserted that a commonly accepted general rule is that 15 common items are adequate.

Still another issue with the common items design is the effect of item scrambling. Scrambling is the rearrangement of the order of items within a form, usually done to discourage cheating. Harris (1991) created four scrambled forms and a base form of an anchor test and equated results across old and new forms via both equipercentile and IRT techniques. She found differences between the base form conversions and scrambled form conversions for all administered forms where up to 50% of the examinees who were administered one of the scrambled forms would have received a different scale score if the base form conversion table was used instead of the scrambled form conversion table. Harris concluded that context, practice and fatigue might have effects on the answers to particular items, which might account for these discrepancies. In a study looking at the effect of item order on the estimates of IRT parameters, Zwirk (1991) discovered that changes in test forms had a substantial impact on the estimation of equating coefficients and that researchers should look closely at the item-parameter invariance assumptions in

this context.

Another concern when using common items is the possibility that the two equating groups will differ systematically in ability (Eignor et al., 1990). This problem is likely to arise when the samples for equating are not randomly selected from the target population of test takers. In a study by Little and Rubin (1994), two forms were administered to biased samples and equated using equipercentile and linear equating methods. They concluded that the two main problems which could occur were bias in the equating function due to the nonrandom selection of the equating groups and excessive variance in the equating function at scores which are under-represented in the population. However, Little and Rubin also concluded that the sample bias may not be a major problem for equating if variance-reducing methods were introduced (e.g., smoothing).

More common though is the belief that differing groups can cause a problem for equating. As Skaggs (1990) pointed out, different samples of examinees who take a form may not produce the same equating function, and this could lead to misinterpretation of the results. Therefore, whenever a single form is administered to dissimilar groups of examinees, there is a potential for lack of invariance. Skaggs found this problem existed for both traditional and IRT equating techniques. Therefore, when conducting a common items design it is important to consider how much the two equating groups differ. As Kolen (1990) noted, for a common items design to be effective, the common items must behave the same in the two groups which are being equated or no equating method will perform well. In his review, Kolen noted that even if the forms were carefully constructed, if the two groups were markedly different even IRT methods could produce

poor results.

This has led many researchers to the idea of matching samples. Lawrence and Dorans (1990) discussed the difference between the use of representative samples and matched samples. Representative sampling occurs when the old-form uses a representative sample of the population who took both a previously equated form of the test (the old form) and the anchor which links it to the new form. The new form sample is a representative sample from the same population who take both the new form of the test and the same common items which link it to the old form. Lawrence and Dorans contrasted this with “matched” samples whereby the new form sample is a representative sample of the population, as just described. The difference here is that the old-form sample is selected from a subpopulation of the old-form group who have taken the old form and the common-item test. The students scores on the old-form test are used as a stratifying variable where a separate sample is selected at each score level on the common-item test, so that the old-form test includes the same number of students at each common-item test score as the new-form sample; thereby forcing the distributions of scores on the common-item test to be the same for both the old and new forms (Lawrence & Dorans, 1990; Livingston et al., 1990)

There is debate as to whether or not matching helps improve equating. In a review of the literature, Skaggs (1990) concluded that matching can be a “risky business.” When studying matching for various equating designs, Kolen (1990) concluded that several studies collectively indicated that matching on the common-item test does not result in more accurate equating. However, Kolen further mentioned that the reviewed studies

focused on common-item test differences as being the main group difference which would affect equating, whereas other differences might have a stronger impact on the equating results. As Kolen pointed out:

...for the anchor test design to be effective, the common items need to behave in the same way in the new group as in the old group. These common items are used to estimate group differences and, based on their relationship with the noncommon items, expand the group differences on the total test forms. If the common items do not behave in the same way in the old and new groups, then no equating method can be expected to function adequately. (p. 100)

Another issue common to all equating techniques is the concept of preequating (Livingston et al., 1990). The item preequating design calls for pretesting items to be included in subsequent forms during the administration of an already equated form. The item statistics for the pretested items are then used to equate scores on the newly constructed forms to the scale used for reporting scores (Kolen & Harris, 1990). However, Kolen and Harris also reported that this method did not perform well in a test of item preequating for equipercentile and IRT equating techniques.

A more subtle problem which can arise when conducting equating is scale drift. Scale drift can occur when equating takes place over time and there is a shift in the composition of the examinee groups. As Petersen, Cook, and Stocking (1983) pointed out, scale drift has happened if the result of equating a new Form "Y" to an old Form "X"

directly is different than the equating results obtained from equating Form “Y” to Form “X” through the intervening forms of “A” and “B”. This process of equating a chain of forms and comparing the results for agreement is often called circular equating (Harris & Crouse, 1993).

Scale drift can occur because the equating has been conducted under non-ideal circumstances, such as differences in the groups taking the form or that the actual test forms used throughout the history of the equating differed in important respects. Scale drift might also occur if the common items do not adequately represent the entire forms (Petersen et al., 1983). In their study, Petersen et al. investigated the effect of scale drift using both linear and IRT equating methods. They concluded that if the forms were relatively parallel, then linear equating techniques worked fine. However, if the forms differed in content and length, then IRT equating using the 3PL model had greater stability.

Equating Issues Associated with IRT

Several issues specific to IRT need to be addressed in IRT equating. An assumption of some IRT models is that the latent trait being measured is unidimensional. However, this may not always be a valid assumption and can present some problems when equating forms via IRT techniques. Dorans and Kingston (1985) suggested that in practice, this strong assumption is likely to be violated. They studied the impact of violating this assumption on IRT equating and showed that dimensionality did have an effect on non Rasch model based equating through its effect on the magnitude of the item

discrimination parameter estimates. Interestingly, they also concluded that these effects were not substantial and that the unidimensional IRT model might be more robust to multidimensionality than expected. Camilli, Wang, et al. (1995) expanded on the Dorans and Kingston study by developing a factor analytic strategy which was consistent with a real multicomponent form containing stand-alone items and “testlets” which allowed them to accurately identify the inter-item correlation structure of the form prior to the equating. These authors agreed with the Dorans and Kingston study that violations of unidimensionality may not have a substantial impact on the true-score conversion tables.

Lautenschlager and Park (1988) discussed the special case of using IRT linking methods for differential item functioning (DIF) studies. They argued that the unidimensionality assumption is suspect any time there is item bias, and that most of the IRT equating procedures are therefore logically flawed for this type of research. In their study, the authors concluded that the item parameter invariance property of IRT is valid only for unbiased items and that the parameter linking methods were adversely affected in the presence of biased items. Hirsch (1989) extended unidimensional IRT form equating to the multidimensional realm by the use of the Multidimensional Item Response Theory Estimation (MIRTE) program. He conducted a common-examinee equating design and found that although he was able to get results comparable to true scores, the ability estimates were unstable. Davey (1996) approached the problem of scaling multidimensional item calibrations. He built upon the more commonly accepted unidimensional linking techniques by using factor analysis techniques to discuss methods which might be used for multidimensional equating.

Another issue which must be considered is scale shrinkage. In a study of IRT vertical equating, Camilli, Yamamoto, and Wang (1993) described how the variance of test scores diminished across time and grade levels. The authors noted that scale shrinkage is generally defined by shrinking variances within and between grade levels, but that they also found some evidence for scale expansion. Scale shrinkage is thought to be a statistical artifact or to come primarily from multidimensionality. The joint maximum likelihood estimation (JMLE) approach toward parameter estimation may cause measurement error and restrict the range of $\hat{\theta}$ (e.g., for all correct or all incorrect response patterns), which could lead to this shrinkage. It is also thought that if the form is multidimensional, the variance of the item parameter and ability estimates will decrease if there is a secondary construct that becomes more influential to performance on later form items (Camilli, Yamamoto, et al. 1993). The authors concluded that the joint maximum likelihood method of parameter estimation is more likely to cause this shrinkage than the actual linking techniques for the forms.

Equating Processes

Traditional Techniques

As mentioned, the two main traditional approaches for equating are the linear and equipercentile methods. These two approaches are both empirically based and rely upon the observed test scores from the two forms to be equated. If the shape of the underlying distributions of the two forms is believed to be the same, then a linear equating method can be used. Linear equating generally tries to solve the equation:

$$I_y(x) = y = \sigma_y \frac{(x - \mu_x)}{\sigma_x} + \mu_y \quad (\text{II.4})$$

where $I_y(x)$ is the linear conversion equation used to convert observed scores (x) on Form X to the scale of Form Y (Kolen & Brennan, 1995). In order for this equation to equate equally well across sample groups, it is assumed that the two forms being equated are equally reliable (Hambleton & Swaminathan, 1985; MacCann, 1990). When using an anchor test (a.k.a., common-item nonequivalent groups) data collection design, two different groups are administered the two different test forms (X and Y), sharing only the common items. However, because the means and standard deviations in the above equation refer to the entire population and not just the two groups sampled, the concept of a synthetic population has been introduced to estimate the parameters (Hanson, 1991; Kolen & Brennan, 1995; MacCann, 1990) described above. Kolen and Brennan (1995) discussed how the two samples can represent two different populations, but that equating needs to be defined for a single population. Therefore, the required parameters can be defined by the following set of equations:

$$\mu_s(X) = w_1\mu_1(X) + w_2\mu_2(X) \quad (\text{II.5})$$

$$\mu_s(Y) = w_1\mu_1(Y) + w_2\mu_2(Y) \quad (\text{II.6})$$

$$\sigma_s^2(X) = w_1\sigma_1^2(X) + w_2\sigma_2^2(X) + w_1w_2[\mu_1(X) - \mu_2(X)]^2 \quad (\text{II.7})$$

and,

$$\sigma_s^2(Y) = w_1\sigma_1^2(Y) + w_2\sigma_2^2(Y) + w_1w_2[\mu_1(Y) - \mu_2(Y)]^2 \quad (\text{II.8})$$

where the w 's are weights and the subscripts 1 and 2 refer to the two nonequivalent

groups being administered the forms. However, in the common-item nonequivalent groups design, each group is administered only one form, so the parameter estimates for that particular group on the other form cannot be estimated directly (i.e., Form X is not administered to group 2 and Form Y is not administered to group 1) (Kolen & Brennan, 1995).

To solve this problem, two major observed score methods have been introduced to express the parameters μ_1 , μ_2 , σ_1^2 and σ_2^2 from directly estimatable parameters. These two approaches are the Tucker and Levine methods (Brennan & Kolen, 1987a; Hanson, 1991; Kolen & Brennan, 1995; MacCann, 1990).

The Tucker method makes two assumptions in order to estimate the parameters. First, it assumes that the regression of both forms on the common items is the same for both populations. The method also assumes that the conditional variance of each form given the common item scores is equivalent for both populations. Given these assumptions, the Tucker method can solve for the intermediate parameters as well as the synthetic population parameters, which are viewed as adjustments to directly observable quantities (Kolen & Brennan, 1995). The Tucker method can be applied for both internal and external item common-item designs.

The Levine method differs from the Tucker method in that it is based upon assumptions of true scores, rather than observed scores. The actual method relates observed scores on Form X to observed scores on Form Y, but the assumptions for the method are related to classical test theory which is a true score model (Feldt & Brennan, 1989; Kolen & Brennan, 1995). The method assumes that there are three true score

equations:

$$X = T_X + E_X, \quad (\text{II.9})$$

$$Y = T_Y + E_Y, \quad (\text{II.10})$$

$$V = T_V + E_V \quad (\text{II.11})$$

where X is an observed score on Form X, Y is an observed score on Form Y and V is an observed score on a common items form. The T and E values are true and error scores on the appropriate forms. This method assumes that the true scores on the Forms X and Y measure the same construct because T_X and T_V and T_Y and T_V both correlate perfectly across the two populations. Because it is assumed that the true score distributions are identical, it can be assumed that X and Y both measure the same construct with a linear relationship between the two scores. Therefore, the true scores of X and Y are assumed to be perfectly correlated, resulting in a congeneric test (MacCann, 1990). The Levine method also assumes that the regression of T_X on T_V is the same linear function for both populations, as is T_Y on T_V . The final assumption in the Levine method is that the measurement error variances for X , Y and V are the same for both populations (Kolen & Brennan, 1995). Further assumptions and restrictions can be applied to this model, resulting in different equations to estimate the parameters (MacCann, 1990). Levine's formula can also be derived for true scores, resulting in slightly different equations to estimate the parameters (Hanson, Zeng, & Kolen, 1993; Kolen & Brennan, 1995).

In a comparison of the Tucker and Levine techniques, Gafni and Melamed (1990) looked at their effectiveness in a circular equating paradigm. They discovered that overall, the Tucker method produced less error when a test was equated to itself with an equating

chain of varying links, but otherwise it resulted in larger cumulative error. MacCann (1990) used these two methods to equate two populations which differed in ability by one-half a standard deviation. He found that the Levine equations gave low values for bias and the RMSE indices, but the Tucker method was unable to cope with these dissimilar populations.

If there is reason to believe that the underlying shapes of score distributions for the two populations are different, then equipercentile equating should be used. This equating method essentially equates the percentile ranks of the two distributions, such that two raw scores from the two distributions are equated if they have the same percentile rank. Once again, if the common-items nonequivalent groups data collection method is chosen, the equipercentile equating function deals with the synthetic population (Jarjoura & Kolen, 1985).

A potential shortcoming of the equipercentile equating method is that it forces agreement at the endpoints of the score ranges for the forms being equated (e.g., the highest scores on both forms will be at the 100th percentile). When using the equipercentile equating method, several other considerations must be made. Among these how to best estimate standard error (Jarjoura & Kolen, 1985; Liou & Cheng, 1995); how to extend the linear equating function to a nonlinear (equipercentile) method (Wang & Kolen, 1996), or whether to use smoothing techniques when conducting an equipercentile equating (Kolen, 1991; Kolen & Brennan, 1995; Zeng, 1995). Smoothing techniques help reduce random error by producing smooth functions. However, when not carefully conducted, it is possible for these techniques to introduce systematic error. Therefore,

ideally smoothing introduces less error than it removes (Kolen & Brennan, 1995).

IRT Methods

If an IRT equating method is chosen, several issues must be addressed. The first decision is which type of data selection design will be employed (Hambleton & Swaminathan, 1985). The most common approach used for IRT equating appears to be the anchor-test or common-item nonequivalent groups design (Cook & Eignor, 1991)

Another issue to be resolved when conducting IRT equating is the choice of the function to represent the ICC. This in turn can influence the choice of statistical procedure selected to perform the equating. Currently, the preferred models are the 1PL, 2PL and 3PL. There is much debate over which of these models to use when performing equating. Some researchers would argue that the 2PL and 3PL models provide more information, and therefore should be used when conducting equating (Baker & Al-Karni, 1991; Stocking & Lord, 1983) while others argue that the Rasch model is more appropriate in that it assures the underlying metric is on an interval scale when the data fit the model (Smith & Kramer, 1992). Also, because fewer item parameters need to be estimated, the Rasch model allows smaller sample sizes than the 3PL model (Lord, 1983).

When conducting vertical equating, the Rasch model does not appear to be as effective as the 2PL or 3PL models. In a review of the literature Skaggs and Lissitz (1986) concluded that most of the research demonstrates the superiority of the 3PL model over the Rasch model, because of the failure of the Rasch model to account for non-zero lower asymptotes. They conceded that the results reviewed did not demonstrate the

consistent superiority of the 3PL model over other IRT models and conventional methods. The authors concluded that there were several likely confounding factors, such as parameter estimation problems, differences in data collection designs and different linking procedures (Skaggs & Lissitz, 1986). In a later study, Skaggs and Lissitz (1988) reaffirmed that the 3PL model generally showed better results for vertical equating than the Rasch model, but that the 3PL model's effectiveness can vary greatly with form length, sample size, estimation and linking strategies, and form content. They also stated that the main problem for using the Rasch model for vertical equating is that under this model samples of examinees at different ability levels produce substantially different equating functions (Skaggs & Lissitz, 1986, 1988). Other authors also believe that the 2PL and 3PL models should be used when conducting IRT equating because they provide more information and better model an actual test (Kolen & Whitney, 1982; Petersen et al., 1983).

Once a model has been chosen, the third issue to be resolved for IRT equating is to obtain parameter estimates for the two forms and put them onto a common scale. Two general approaches for doing this are concurrent estimation and scale linking. When a concurrent estimation is conducted, all items from the two forms are pooled and the θ and item parameter estimates are computed in one data pass (Baker, 1984; Kolen & Brennan, 1995; Wingersky & Lord, 1984). This approach is not as useful for putting newly calibrated items onto a current scale defined by an existing item set (Davey, 1996).

A more common way of putting the parameters of the two forms onto the same metric is through scale linking, which exploits the fact that the parameter estimates from

the two forms are linear transformations of each other. This follows directly from the invariance principle of IRT which states that the item parameter and ability estimates are independent of the actual sample form in which common items are embodied. However, the item and trait parameter estimates from different samples are likely to be on different metrics because the origin, and usually the unit scale, of the trait's metric cannot be uniquely determined (Davey, 1996). That is, the item response functions $P_i(\theta_j)$ are usually treated as functions of $a_i(\theta_j - b_i)$ (Lord, 1980). By adding a constant to every θ_j and b_i value, the quantity $a_i(\theta_j - b_i)$ and the response function, $P_i(\theta_j)$, remain unchanged. The same is true for multiplying every θ_j by a constant and dividing every a_i by the same constant. Typically, to resolve this scale indeterminacy the mean and standard deviation of θ are fixed to 0 and 1 (Baker, 1990; Lord, 1980; Mislevy & Stocking, 1989). Imposing these constraints resolves the indeterminacy of scale and allows the parameters to be estimated (Davey, 1996). However, if item parameters are estimated separately for two forms, the actual values of the estimates will differ because the scales established for the two calibrations will differ in origin and unit of measurement. Fortunately, the relationship between the two scales is linear and estimates of item difficulties and examinee abilities can be defined by the following equations (Stocking & Lord, 1983):

$$\hat{\theta}_y = \alpha \hat{\theta}_x + \beta \quad (\text{II.12})$$

$$b_y = \alpha b_x + \beta \quad (\text{II.13})$$

and

$$\alpha_y = \alpha_x / \alpha \quad (\text{II.14})$$

where

$\hat{\theta}_y$ is the ability estimate on the Form Y, or target metric

$\hat{\theta}_x$ is the ability estimate on the Form X, or initial metric

α_y = the discrimination parameter for a particular item on the target metric

α_x = the discrimination parameter for a particular item on the initial metric

b_y = the difficulty parameter for a particular item on the target metric

b_x = the difficulty parameter for a particular item on the initial metric

and α and β are the slope and intercept of the equating line.

These equations describe the basic transformation among metrics (Baker, 1992, 1993, 1996; Hambleton & Swaminathan, 1985). For the 1PL model the α slope value is a constant. This implies that the ability estimates merely differ by a constant. For the 2PL and 3PL models, the relationships between the α 's and b 's can be used to determine the equating coefficients. In the case of a 3PL model, the c parameter is not transformed because its value is independent of the θ metric (Baker, 1992).

Once the item parameter estimates have been obtained for the two forms both sets of estimates are placed on the same scale through equations (II.12 - II.14). The values for α and β may be obtained by one of four different strategies; 1) regression method; 2) "Mean and sigma" procedure; 3) Robust "mean and sigma" procedure (Linn, Levine, Hastings & Wardrop, 1981); and 4) test characteristic curve (TCC) method (Stocking & Lord, 1983).

In the regression method, the target metric is regressed on the initial metric in a linear regression. The values for the α and β coefficients are calculated as the slope and y-intercept of the regression line, respectively. Unfortunately, the relationship will not necessarily be symmetric. That is, the values obtained for α and β can be different, depending on which form is chosen as the base form in the regression. In addition, the errors are not necessarily identically distributed because each item and ability parameter estimate has a separate standard error of estimate. However, with the Rasch model the regression method can be appropriately employed because the slope estimate would result in a symmetric relationship (Hambleton & Swaminathan, 1985).

The mean and sigma method is an extension of the regression method for finding the relationship between two scales. This method uses the linear relationship between the estimated b values for the two forms discussed above such that the mean and standard deviation of the transformed distribution of estimated b 's from the initial form's calibration are equal to the mean and standard deviation of estimated b 's from the target form's calibration (Hambleton & Swaminathan, 1985; Stocking & Lord, 1983). The mean and sigma method makes use of the equations:

$$\alpha = s_{b_y} / s_{b_x} \quad (\text{II.15})$$

and

$$\beta = \bar{b}_y - \alpha \bar{b}_x \quad (\text{II.16})$$

to compute the coefficients α and β used in equations II.12 - II.14.

However, even though the mean and sigma method is symmetric, it does not take

into account the affect of outliers or that each item and ability parameter may be estimated with varying degrees of accuracy. The robust mean and sigma method has been developed to account for this varying degree of estimation accuracy by taking the individual error terms into account. This method essentially weights the estimates inversely by their variances. Therefore, those estimates with larger variances will receive smaller weights.

For the robust mean and sigma method a weight is established for each (b_{xj}, b_{yj}) pair so that $w_j = \max[\text{var}(b_{xj}), \text{var}(b_{yj})]$ where the variances are associated with the j^{th} b . The weights are then scaled by the equation:

$$w'_j = w_j / \left(\sum_{j=1}^n w_j \right) \quad (\text{II.17})$$

after which the transformed b values are computed such that:

$$b'_{xj} = w'_j b_{xj} \text{ and } b'_{yj} = w'_j b_{yj} \quad (\text{II.18})$$

Using the b'_x 's and b'_y 's, new values for the means and standard deviations of the weighted scores are computed. These means and standard deviations are then substituted into equations II.15 and II.16 for the mean and sigma method so that α and β can be computed as before.

While an improvement over the mean and sigma approach, this method still does not take into account the effect of outliers. A modified version of this robust mean and sigma method has been proposed (Stocking & Lord, 1983). This modification computes Tukey weights based upon the perpendicular distance of each point to the calculated

equating line (Equation II.13). These new weights are used to re-weight each (b'_{xj}, b'_{yj}) point and this new information is then used to solve equations II.15 and II.16. This approach has the advantage of giving low weights to both poorly estimated parameters and to outliers.

All three preceding approaches only use the information contained in the item difficulties (i.e., $b_{yj} = \alpha b_{xj} + \beta$) to establish the equating line. In addition to this relationship among the b 's, there is the additional relationship between the items α 's on the two forms, $a_{yj} = a_{xj}/\alpha$. The TCC method proposed by Haebara (1980) and Stocking & Lord (1983) was designed to use the relationship between the discrimination estimates for an item as well as its difficulty parameter estimates. The important distinction of the TCC method is that it takes into account all available information (i.e., TCC item difficulties and discrimination values for all of the items, rather than just summary statistics). This method assumes that if the item parameters for an ICC are known, then the true score for any examinee will be the same for the two different calibrations of the same test, provided the scales have been correctly linearly transformed (Baker & Al-Karni, 1991).

The constants, α and β , are chosen to minimize the difference between the true score estimates $\hat{\xi}_j$ and $\hat{\xi}'_j$ through minimizing an appropriate function. Stocking &

Lord (1983) suggested minimizing the quadratic loss function:

$$I' = \frac{1}{N} \sum_{j=1}^N \left(\hat{\xi}_j - \hat{\xi}'_j \right)^2 \quad (\text{II.19})$$

where N is the number of examinees in the arbitrary group, $\hat{\xi}_j$ was defined earlier and

$\hat{\xi}_j'$ represents the estimated true score obtained from the transformed metric (see III.10).

This function is minimized when $\frac{\partial F}{\partial \alpha} = \frac{\partial F}{\partial \beta} = 0$. This particular function is solved

by a system of simultaneous equations, which does not have a closed form (Baker, 1996).

The Stocking and Lord (1983) approach uses an iterative multivariate search technique.

Baker, Al-Karni, and Al-Dosary (1991) solved for the constants in this model by using the

Davidon-Fletcher-Powell technique. In order to evaluate F , $\hat{\xi}_j$ and $\hat{\xi}_j'$ are evaluated for

an arbitrary set of points along the ability scale. The EQUATE program (Baker, 1996)

implements this solution and computes the α and the β equating coefficients and provides

the value of the loss-function F at these values. The program also transforms the item and

ability parameter estimates from one metric to the other (Baker, 1996).

Stocking and Lord (1983) compared the transformations obtained from the TCC

method to the robust mean and sigma method on over 20 pairs of forms. They showed

that in all cases the robust mean and sigma method never provided a better fit to the

estimated item difficulties and discriminations. From this study Stocking and Lord

concluded that the TCC method is logically superior in that it uses more available

information for the scaling. Hambleton and Swaminathan (1985) recommended that the

TCC method be used when working with the 2PL and 3PL models. Kolen and Brennan

(1995) concurred that the TCC method should be used as well as mean sigma methods when IRT is used for scale transformation.

Baker and Al-Karni (1991) concluded that the Stocking and Lord (1983) procedure was the de facto standard with which other procedures for computing equating coefficients needed to be compared against. They compared the TCC method to an approach by Loyd and Hoover (1980) which used the ratio of the estimated α 's obtained from the two calibrations to obtain the slope information. They found that the Stocking and Lord procedure was less sensitive to atypical combinations of the underlying ability, item difficulty and discrimination values (e.g., low-ability, low-discrimination and high-difficulty) than the Loyd and Hoover method. They concluded that the Stocking and Lord procedure should be used when the researcher believes that the calibration may be associated with troublesome data sets. Baker extended the TCC method to the graded response model (Samejima, 1969, 1972) and to the nominal response model (Bock, 1972) (see Baker, 1992, 1993).

One criticism of the TCC method proposed by Stocking and Lord (1983) is that it does not take into account the error in estimating item parameters (Divgi, 1985; Kim & Cohen, 1995). Kolen and Brennan (1995) argued that this might not be crucial when there is a large sample size and well estimated item characteristic curves. However, these authors also pointed out that situations might arise where the parameter estimate error might be a problem (e.g., disproportionate sample sizes used for the sample forms) and that more work needed to be done in this area.

Measuring the Accuracy of Equating Results

Of great concern when equating forms is the question of how to judge the accuracy of the equating results (Kolen & Harris, 1990). As Kolen and Brennan (1995) noted, there are two main sources of error when equating: random error and systematic error. Random error in equating results from the fact that samples have been used for the equating relationship. Whenever samples are used to represent a population, there will be random error associated with it. Systematic error would be introduced in the equating relationship when the equating estimation method introduces bias in estimating the relationship, or if the statistical assumptions required for the equating relationship are violated. Systematic error might also occur if the sampling design is flawed or incorrectly implemented. Most standard error estimation techniques attempt to measure the random error associated with the equating procedure (Kolen & Brennan, 1995).

Kolen and Brennan (1995) defined the standard error of equating as “the standard deviation of equated scores over hypothetical replications of an equating procedure in samples from a population or populations of examinees” (p. 211). This standard error can be estimated through a bootstrap procedure (Efron, 1982; Efron and Tibshirani, 1993) where multiple random samples with replacement are drawn from the sample data. The bootstrap method can be very effective, but has the drawback of being computationally intensive.

Another technique to estimate standard error is the delta method, where an approximate standard error can be derived as a function of other standard error estimates for closely related statistics. This method follows the steps of 1) specifying the error

variances and covariances for each parameter estimate; 2) finding the partial derivative of the equating equation with respect to each parameter estimate; and 3) substituting the variances and partial derivatives into a Taylor series expansion (Kolen & Brennan, 1995).

There are many other standard error estimation techniques with the choice often being made on the basis of the equating design and the method used. When using IRT equating techniques, various methods exist to evaluate the error associated with estimating and calibrating the item and ability parameters. Liou (1990) discussed the use of the standardized mean-square difference approach in estimating the errors of the α , b , c , and θ parameters for three IRT scaling procedures. She defined this value as the ratio between the two quantities,

$$\frac{\sum_i^m (\omega_i - \hat{\omega}_i)^2}{m} \text{ and } (s_{\omega}^2 + s_{\hat{\omega}}^2)/2 \quad (\text{II.20})$$

where

ω = the theoretical parameter

$\hat{\omega}$ = the parameter estimate

s^2 = the variance estimates and

m = the number of parameters in the test.

Another commonly used method for determining the accuracy of equating results is the root mean square (RMS) approach (Harris & Crouse, 1993). They defined RMS as:

$$RMS = \left(\frac{\sum_{k=1}^K f_k (A_k - B_k)^2}{\sum_{k=1}^K f_k} \right)^{1/2} \quad (II.21)$$

where

A_k = the equated score for raw score k on the target form

B_k = the true or criterion score

f_k = the frequency of a raw score of k on the target form, and

k = a raw score on the target form (running over K possible raw scores).

Harris and Crouse (1993) also described the mean absolute difference (MAD), the mean signed difference (MSD), and the standardized root mean square difference (SRMSD) approaches. They defined them as:

$$MAD = \frac{\sum_{k=1}^K f_k |A_k - B_k|}{\sum_{k=1}^K f_k} \quad (II.22)$$

$$\text{MSD} = \frac{\sum_{k=1}^K f_k (A_k - B_k)}{\sum_{k=1}^K f_k} \quad (\text{II.23})$$

$$\text{SRMSD} = \left(\frac{1/n \sum_{k=1}^K (A_k - B_k)^2}{(S_A^2 + S_B^2)/2} \right)^{1/2} \quad (\text{II.24})$$

where

A_k, B_k and f_k are the same as defined before

n = the number of score points

S_A^2 = the variance of A

S_B^2 = the variance of B.

Summary

It can be seen that there are many different ways to define and implement equating. While all equating techniques have the core purpose of statistically adjusting scores on different forms to make the scores interchangeable, each approach has its own set of assumptions, properties, and limitations. The choice of an equating technique is based

upon several factors such as underlying theoretical model, data collection design, and whether horizontal or vertical equating is desired. Careful consideration must be made of each of these factors before undertaking an equating project so that correct analyses and interpretations can be made. This review of the literature would suggest that it is especially important to test the underlying assumptions associated with equating techniques and measure the accuracy of the results obtained.

There are still several issues in equating which need to be addressed. For example, when using IRT equating techniques, the coefficients for the equating line are estimated, based upon item parameter estimates. An implied assumption is that the parameters were well-estimated. However, there is always a certain amount of error associated with estimating item parameters. What type of effect would error in the item parameter estimates have on the calculation of the equating coefficients and the estimation of a person's true score? This issue is one of many which requires further research.

Chapter III - Methodology

Statement of Problem

The TCC method as discussed by Stocking and Lord (1983), uses the information available for all $\hat{\theta}$ s to minimize the difference between the estimated true scores on the initial and the target metrics (i.e., to match the TCC's of the two forms) through a loss-minimization algorithm. This loss-minimization algorithm produces the equating coefficients (α and β) for equations (II.12 - II.14).

However, a potential problem arises in that the equating coefficients estimates are calculated using item parameter estimates, not the item parameters. Because the estimates for the equating coefficients are based upon estimates of item parameters, those conditions which affect the item parameter estimates could be expected to influence the equating coefficients estimates. Several studies have been conducted which have studied the factors associated with obtaining accurate item parameter and ability estimates. Yet those studies that look at the accuracy of equating tend to assume that the item parameter estimates used to obtain the equating coefficients estimates are accurate (e.g., Harris & Crouse, 1993). The robustness of the TCC method for obtaining equating coefficients (linking) and computing estimated true scores (equating) when there is error in the item parameter estimates has not been studied.

Purpose and Overview of study

This simulation study looked at the accuracy and robustness of the TCC method for estimating equating coefficients when there is error in the item parameter estimates. This study modeled a horizontal internal anchor equating situation and assumed that the

only difference between the common item parameter estimates was the error associated with them. Item parameter “estimates” for the initial and target metrics were generated from sampling distributions with known standard errors. The estimates for the initial metric were equated to the target metric using the TCC method. For each set of conditions, the recovery of known equating coefficients was examined and estimated true scores were compared to the known true scores. In addition, an Improvement Ratio was introduced that expresses the loss function F (II.19) as a common “improvement” indicator.

Determination of standard error for item parameter estimates

There are several factors that can influence the standard error of item parameter estimates. It is known that sample size and the choice of an ICC function are related to the standard errors as is the item parameter value (i.e., extreme item parameter values usually have larger standard errors). Traditionally, most IRT equating simulations require the generation of sample data based on a particular ICC function. The generated data are then used to estimate the item parameters and ability estimates for the simulation. However, because this study was concerned about the effect of error in these estimates, more direct control over the standard error was desired. This study directly manipulated the standard error for the item parameter estimates.

For this study horizontal equating was based on the 2PL and 3PL models. The amount of standard error associated with the item parameter estimates was set at three levels (see Tables 1 - 4). Because we would expect larger standard errors for item

parameter estimates with a 3PL function these values were correspondingly higher than those of the 2PL.

Table 1 - 2PL Standard Errors (SE) - Non-related error

Item Parameter	Low level of SE	Moderate level of SE	High Level of SE
<i>a</i>	.07	.15	.38
<i>b</i>	.11	.25	.63

Table 2 - 3PL Standard Errors (SE) - Non-related error

Item Parameter	Low level of SE	Moderate level of SE	High Level of SE
<i>a</i>	.10	.23	.53
<i>b</i>	.30	.91	2.73
<i>c</i>	.05	.10	.20

The low level of standard errors for the *a* and *b* parameters are associated with a sample size of $N=2500$ (Thissen and Wainer, 1982). The moderate level of standard errors for these parameters are associated with a sample size of $N=500$. The standard errors calculated for these two sample sizes are the minimal asymptotic standard errors that would be expected in practice. These values assume that there is model-data fit and that all other IRT assumptions hold. In practice, there are likely to be many cases of parameter estimation where the associated standard error is much higher than the moderate level. Therefore, because the moderate level of standard errors is approximately 2.5 to 3 times the low level, these ratios are used with respect to the moderate level to obtain the third and highest level of standard error. The low, moderate and high levels of

standard error for the c parameter associated with the 3PL model is from R. J. De Ayala (personal communication, December 12, 1997).

In addition, this study modeled a second type of SE (the “related error case”) which allowed the SE to be dissimilar across the a and b parameter estimates such that the more extreme b values had higher standard error than the b values in the middle of the scale. In practice it would also be expected that those items which have b 's in the center portion of the difficulty scale would have larger a 's than those items at either extreme of the difficulty scale. Additionally, It would be expected that very difficult items would have a higher guessing parameter associated with them and very easy items would have a lower guessing parameter associated with them. These relationships among the parameters were modeled in this simulation for the related error conditions. The values of SE attached to a and b for the related error case were as defined in Tables 3 and 4.

Table 3 - Standard Errors for b when SE related to b value

ICC Function	Low Level of SE	Moderate Level of SE	High Level of SE
<i>when b is:</i>	-3, -1.5, 0, 1.5, 3	-3, -1.5, 0, 1.5, 3	-3, -1.5, 0, 1.5, 3
2PL	.22, .11, .04, .11, .22	.48, .25, .10, .24, .48	1.2, .63, .25, .63, 1.2
3PL	1.53, .60, .22, .14, .41	3.41, 1.34, .49, .31, .91	7.5, 2.95, .11, .68, 2.0

Table 4 - Standard Errors for a when SE related to b values

ICC Function	Low Level of SE	Moderate Level of SE	High Level of SE
<i>when b is:</i>	-3, -1.5, 0, 1.5, 3	-3, -1.5, 0, 1.5, 3	-3, -1.5, 0, 1.5, 3
2PL	.09, .07, .05, .07, .09	.20, .15, .12, .15, .20	.50, .38, .30, .38, .50
3PL	.18, .14, .13, .20, .32	.40, .31, .29, .45, .71	.88, .68, .64, .99, 1.56

Selection of item parameter estimates

This study directly sampled item parameter estimates from sampling distributions with known standard error. Because the main purpose of this study was to examine the effect of error in the item parameter estimates on the TCC equating process, directly sampling items from a known distribution provided more direct control over the standard error of any given item parameter. Furthermore, because this study simulated a horizontal equating situation, the expected equating coefficients α and β were 1 and 0, respectively. This implies that the only difference between common item parameter estimates on different forms would be associated with error, and unless there is bias in the estimation procedure, this error should be random. The loss function, F , is estimated across all common items with the loss minimized for the entire common item set (this value is 0 when the test characteristic curves of the two forms are identical). Therefore, it would be expected that the error associated with any individual item is essentially eliminated by other item parameter estimates among the common items. By directly sampling item parameter estimates from distributions with known standard error, the only difference between any two estimates from one distribution would be random error with cancellation among error terms to be expected across the multiple items, as discussed.

Estimation of equating coefficients

The EQUATE program (Baker, 1991, 1993) was used to obtain the equating coefficients. Whereas Stocking and Lord (1983) originally proposed using all information available for all examinees in the estimation of equating coefficients, the EQUATE

program uses specified points along the θ scale. Specifically, evenly spaced ability points along the θ scale are chosen and the loss function is evaluated at these points. In several tests, Baker (1991) showed that this approach was robust and less time-consuming to compute the α and β estimates. If more points are evaluated, it would be expected that the equating would be better. Baker recommended the use of 11 or 21 points. However, the EQUATE program will allow the user to specify up to 100 points for estimation. For practical purposes, most researchers will choose to maximize the number of estimation points to obtain the most accurate coefficient estimates possible. For this study, the θ scale was set to range from -4 to 4 as discussed by Baker (1992) and Kim and Cohen (1995) and the number of evaluation points was 99.

Variables in the study

Independent variables

1. Amount of estimation error in item parameter estimates. This was the main variable of interest. There were three levels of this variable (Low, Moderate, High) as defined in Tables 1 - 4. The standard errors for each item parameter estimate were held constant across the two forms being equated.
2. Choice of ICC function. This study modeled both the 2PL and 3PL cases. As noted, the 3PL simulations had higher associated error than the 2PL simulations.
3. The number of common items between forms. It was expected that the number of items in common between the two forms would affect the estimation of the equating coefficients. There were three levels of the number of common items factor, 5 items, 15 items and 25 items. In order to represent the entire form the

common items were sampled from the entire b range of the form.

4. Relationship of standard error with a and b estimates. This study looked at two levels of association. The first level is referred to as the “non-related error case” and held the SE associated with the a and b parameter estimates approximately the same across all item parameter estimates. The values of SE attached to b for the non-related error case was set to those values defined in Tables 1 and 2. The second level (the “related error case”) allowed the SE to be dissimilar across the α and b parameter estimates as reflected in Tables 3 and 4.

Dependent variables

1. Accuracy of estimation for the equating coefficients (α and β) as measured by the RMSE and Bias terms between the actual equating parameters and the estimated parameters. Because this study simulated a horizontal equating situation the equating coefficients α and β were 1 and 0, respectively. RMSE and Bias were calculated between these equating coefficients and their estimates as follows:

$$\text{RMSE} = \sqrt{\frac{\sum (\hat{\lambda} - \lambda)^2}{N}} \quad (\text{III.1})$$

$$\text{Bias} = \frac{\sum (\hat{\lambda} - \lambda)}{N} \quad (\text{III.2})$$

where λ and $\hat{\lambda}$ are the values for the coefficients (i.e., $\lambda =$ values of 1 and 0) and their estimates, respectively, and N is the number of replications.

2. The average value of F (the loss-minimization function employed by the EQUATE program's implementation of the TCC method). This value is 0 when the test characteristic curves of the two forms are identical.
3. Improvement Ratio (IR). This is the ratio of the F obtained from a converged equating solution to the maximum F possible for the equating. That is,

$$IR = \frac{F_{\max} - F_{\text{converged}}}{F_{\max}} \quad (\text{III.3}).$$

F_{\max} was obtained by estimating the F value using the item parameter estimates of the initial and target forms prior to equating. F_{\max} represents the worst case scenario of F for a given data set. It is the maximum size of the loss function for a pair of forms being equated. $F_{\text{converged}}$ represents the best obtained case for a pair of forms. It is the minimum size of the loss function for the forms being equated. Therefore, the numerator of IR represents the difference between the worst possible linking for a pair of forms and the best obtained linking.

For the 3PL case, the following equations were used:

$$F_{\max} = \frac{1}{G} \sum_{g=1}^G \left[\sum_{i=1}^m P_i(\theta_g) - \sum_{i=1}^m P_i(\theta_g) \right]^2, \quad (\text{III.4})$$

$$P_i(\theta_g) = \hat{c}_{init_i} + \left(1 - \hat{c}_{init_i}\right) \frac{e^{\hat{a}_{mit_i}(\theta_g - \hat{b}_{mit_i})}}{1 + e^{\hat{a}_{mit_i}(\theta_g - \hat{b}_{mit_i})}} \quad (\text{III.5})$$

and

$$P_i'(\theta_g) = \hat{c}_{igt_i} + \left(1 - \hat{c}_{igt_i}\right) \frac{e^{\hat{a}_{igt_i}(\theta_g - \hat{b}_{igt_i})}}{1 + e^{\hat{a}_{igt_i}(\theta_g - \hat{b}_{igt_i})}} \quad (\text{III.6})$$

where

G = the number of estimation points along the θ scale used for the equating

m = the number of common items

\hat{a}_{mit_i} , \hat{b}_{mit_i} , and \hat{c}_{mit_i} are item parameter estimates for the initial form,

\hat{a}_{igt_i} , \hat{b}_{igt_i} , and \hat{c}_{igt_i} are item parameter estimates for the target form,

and θ_g represents a particular θ value at an estimation point along the θ scale (-4 to

4). For the 2PL case, the c values will be 0.

$F_{converged}$ was obtained from the EQUATE program for a converged solution. For the 3PL case, this value is defined by:

$$F_{converged} = \frac{1}{G} \sum_{g=1}^G \left[\sum_{i=1}^m P_i(\theta_g) - \sum_{i=1}^m P_i^*(\theta_g) \right]^2, \quad (\text{III.7})$$

$$P_i^*(\theta_g) = \hat{c}'_i + (1 - c'_i) \frac{e^{\hat{a}'_i(\theta_g - \hat{b}'_i)}}{1 + e^{\hat{a}'_i(\theta_g - \hat{b}'_i)}} \quad (\text{III.8})$$

where $P_i^*(\theta_g)$ is as defined in III.6, G and m are as defined above and \hat{a}' , \hat{b}' ,

and \hat{c}' are the equated item parameter estimates from a converged solution.

Again, for the 2PL case c is 0.

IR allows for direct comparisons across different pairs of forms. The ratio has a range from 0 to 1 where a higher value indicates better equating.

4. The average RMSE for estimated true scores. Whereas the preceding dependent variables were designed to measure the accuracy of the linking process, this dependent variable was designed to assess whether there are any practical effects on the equated ability estimates. The true scores were obtained using ability and item parameters such that

$$\xi_j = \sum_{i=1}^n \left[c_i + (1 - c_i) \frac{e^{a_i(\theta_j - b_i)}}{1 + e^{a_i(\theta_j - b_i)}} \right] \quad (\text{III.9})$$

where n = the test length. The estimated true scores were based on the transformed ability and item parameter estimates such that

$$\hat{\xi}'_j = \sum_{i=1}^n \left[\hat{c}'_i + (1 - \hat{c}'_i) \frac{e^{\hat{a}'_i(\hat{\theta}'_j - \hat{b}'_i)}}{1 + e^{\hat{a}'_i(\hat{\theta}'_j - \hat{b}'_i)}} \right] \quad (\text{III.10})$$

for the 3PL cases (and $c = 0$ for the 2PL cases).

Procedures for study

The crossing of the standard error factor (low, moderate, high), the number of common items factor (5, 15 or 25), the ICC function factor (2PL or 3PL) and the relationship of the standard error with the b estimates (non-related error or related error) produced 36 cells. 250 replications were conducted for each cell. There were two major components to this study, the linking portion and the equating portion.

The simulation used the EQUATE program (Baker, 1991, 1993) and original software written by the author (see Appendix L). The original software was written in the C programming language (Microsoft, 1997). For the simulation the ran1 algorithm from Press, Teukolsky, Vetterling, and Flanner (1996) was modified such that all seeds and values were saved to a file for subsequent executions of the program as suggested by Baker (1997). This eliminated any possible periodicity in the random numbers across the cells of the study. The program was initially seeded with a value of -100415.

Linking study

This part of the study looked at the accuracy and robustness of the equating process by examining α , β , F , and the IR for each cell. The procedures for implementing this part of the study were as follows.

Step 1: Parameter Generation

Two sets of item parameters (non-related parameters and related parameters) were

generated for this simulation. There were 18 cells for each of these conditions with the same set of item parameters used for all 18 cells within a set (i.e., all non-related error conditions used a common item parameter set and all related error conditions used a second common item parameter set). Item parameters for fifty items were created for each set of conditions (refer to Appendix B for the distributions of these parameters). For both the non-related and related error conditions the a , b , and c item parameters were randomly sampled from a Lognormal[1.25, .09], a Normal[0,1] and a Uniform[0, .25] distribution, respectively. In practice, several studies have found that it is not practical or necessary to model the covariance between the a and b parameters (e.g., deGruijter, 1984; Kim & Cohen, 1995; Lord, 1982) and so these values were sampled independently.

The original proposal called for \hat{a} s to model the literature (Skaggs & Lissitz, 1988) and be sampled from a Lognormal[.8, .01] distribution. However, preliminary tests which used this distribution showed that the values for a (and correspondingly \hat{a}) had a very narrow range and the \hat{a} estimates did not exhibit a lognormal distribution.

The a parameter distribution was then empirically modified to find a more suitable (i.e., realistic) distribution by setting the mean of the sampling distribution to 1.25 and increasing the variance until a suitably shaped distribution of a parameters was created. This resulted in using a Lognormal[1.25, 0.9] sampling distribution that was used in this study.

Similarly, the original Standard Error values reported for the c parameter were derived from Thissen and Wainer (1982). However, pilot testing of this study revealed that a floor/ceiling effect of the c parameters was quickly encountered in the moderate and

high levels of SE, resulting in unusable parameters. Again, the decision was made to base the SE levels for the c parameters upon more recent literature (R. J. De Ayala, personal communication, December 12, 1997) which empirically modeled these values. It is these latter values of SE for the c parameter which were used in this study.

Step 2: Item Generation.

Two forms, an initial and target form, were created. Each "form" consisted of 50 items. Because the simulation was modeling a horizontal test equating, the item parameters for the 50 items were the same for each form. Once the item parameters were selected, a sampling distribution was created for each item parameter where the mean of the distribution was set equal to the item's parameter value and the standard deviation was set equal to the desired standard error for that particular condition. Two item parameter estimates were then randomly selected from each sampling distribution, one for the target form and one for the initial form. This process was repeated until item parameter estimates were selected for all 100 items (50 for the target and 50 for the initial forms). The same item parameters were used for all replications within a cell.

For the related error conditions the b parameter was sampled first. This value was then examined to determine if it was beyond an upper or lower boundary (1.5 and -1.5, respectively). If the sampled b value exceeded one of these boundaries the α parameter was sampled until it was less than a minimum α value of 1.0. If the sampled b value was between these boundaries, the α parameter was sampled until it was ≥ 1.0 . This forced the more extreme b values to have lower discriminations than the b values in the middle of the scale.

In addition, if the sampled b value exceeded the upper or lower boundary it was classified as either a difficult or an easy item and the c parameter was sampled until it was correspondingly higher ($\geq .17$) or lower ($\leq .08$) to correspond to the difficulty of the item. This procedure was repeated for all fifty items in this parameter set.

The creation of the related items parameter set allowed for related item estimates to be created for each related-error cell. The value of the b parameter for a particular item was used to determine the standard errors for the for the a and b parameter estimates (see Tables 3 and 4). In this way unique item parameter estimates could be created for replication.

Step 3: Common Item Selection.

Initial and target forms were created with 50 items each and a 1 to 1 correspondence of the items between the forms (e.g., the first item on the initial form corresponded to the first item on the target form). Depending upon the condition, either 5, 15 or 25 items in each form were chosen as common. These items were selected to represent the range of the b item parameter estimates generated in the second step. The items on both the initial and target forms were sorted into ascending order by the b estimate on the target form and the appropriate number of common items were then chosen from across the range of items. Starting from the lowest \hat{b} value, every second (25 common items), every third item (15 common items) or every twelfth item (5 common items) was selected. It should be noted that for the 15 common item factor the sampling

interval alternated between 3 and 4 in order to provide complete coverage of the \hat{b} range including the extreme positive end of \hat{b} . The common items were uniquely chosen for each replication.

F_{\max} was computed from these two sets of common items. The number of evaluation points along the θ scale was 99 for the F_{\max} calculation.

Step 4: Estimate the equating coefficients and calculate IR

Once the item parameter estimates were obtained for both the initial and target forms, the common items chosen in Step 3 on the initial metric were equated to the target metric. This step was accomplished by using the EQUATE program (Baker, 1991, 1993); 99 evaluation points were used in order to obtain the most accurate conversion possible. Once convergence was obtained the estimates of α and β were used in the appropriate RMSE and Bias equations. In addition, the IR was calculated for each replication within a cell and the average IR was obtained for each cell.

Equating study

After the initial and target metrics were linked, it was possible to obtain estimated true scores by using the transformed ability and item parameter estimates in equation III.10. This portion of the study was conducted to assess what type of practical effects the linking had on the equated ability estimates. This equating portion of the study was accomplished by the following method.

Step 1: Response Data Generation

In order to obtain estimated true scores, response data were needed. 1000 normal deviates were randomly sampled from a $N(0,1)$ distribution and used as the θ_j s for the simulation. The probability of a correct response to an item was calculated by using the appropriate model (i.e., 2PL or 3PL where the value for D was set to 1) and corresponding item parameters. The $P_i(\theta_j)$ obtained for each item was then compared against a randomly drawn number from a Uniform[0,1] distribution. If the $P_i(\theta_j)$ value was greater than the random number, the answer to that item was coded a 1 (correct), otherwise it was coded a 0. This process was repeated for each of the 50 items on the initial form to create a response vector for each of 1000 simulees. The same response data were used to estimate the ability parameters for each replication within a cell.

Step 2: Estimation of Initial Form $\hat{\theta}_{init}$ s and Transformation to $\hat{\theta}'$ s

After the response data were generated for 1000 simulated examinees it was used to estimate θ_j ($\hat{\theta}_{j_{init}}$) for each examinee using the initial form item parameter estimates and the appropriate model (i.e., 2PL or 3PL). Estimation of $\hat{\theta}_{j_{init}}$ s was accomplished by using the expected a posteriori (EAP) algorithm (Bock and Mislevy, 1982). This study used 15 quadrature points between -4.0 and 4.0 (De Ayala, Schafer, and Sava-Bolesta, 1995) and used Bock and Mislevy's suggestion to assume a normal prior distribution and use weights equal to the prior discrete probability at these points (see Table 5). The

equating coefficient estimates obtained in the linking portion of the study were then used to transform these estimated $\hat{\theta}_{j_{init}}$ s from the initial metric to $\hat{\theta}'_j$ s in the transformed metric.

Table 5
Quadrature values and weights used in EAP estimation

Quad Number	Quad Value	Weight
1	-4.0000	0.0001
2	-3.4286	0.0006
3	-2.8571	0.0038
4	-2.2857	0.0167
5	-1.7143	0.0524
6	-1.1429	0.1186
7	-0.5714	0.1936
8	0.0000	0.2280
9	0.5714	0.1936
10	1.1429	0.1186
11	1.7143	0.0524
12	2.2857	0.0167
13	2.8571	0.0038
14	3.4286	0.0006
15	4.0000	0.0001

Step 3: Estimation of True Scores

A true score for each simulee was computed using the generated θ_j s and item parameters. Then, for each of the 1000 $\hat{\theta}'_j$ s estimated true scores were obtained using the transformed item parameter estimates (\hat{a}' , \hat{b}' , and \hat{c}') obtained in the linking stage and equation III.10. (The EQUATE program provides \hat{a}' , \hat{b}' , and \hat{c}' by using the α and β estimates and the linear transformation equations (II.12-II.14).) The RMSE of the estimated true scores was computed for each replication and an average of the RMSE's

for the 250 replications was calculated for each cell in the study.

Summary

This study looked at the accuracy and robustness of the TCC method for estimating equating coefficients when there was error in the item parameter estimates by simulating a horizontal internal anchor equating situation. Four independent variables; amount of estimation error in item parameter estimates, choice of ICC function, number of common items between forms, and relationship of standard error and b estimates, were fully crossed for analysis.

The analysis was divided into two components. The linking portion studied the accuracy of the equating coefficients estimation. In order to analyze this accuracy and robustness RMSE and Bias were calculated for α and β for each of the cells. In addition, the F values obtained from the equating were evaluated and an Improvement Ratio was introduced to represent how well the test characteristic curves were matched for any two particular data sets.

The second component of the analysis was to estimate true scores using transformed ability and item parameter estimates obtained from the linking. These true score estimates were compared against the known true scores and the RMSE was calculated to determine the accuracy of the true score estimates. For a summary of this study's design choices and its associated references see Appendix A.

Chapter IV - Results

Nomenclature

This study consisted of 36 separate cells with 250 replications per cell. The cells were fully crossed on the variables Amount of Estimation Error (L, M, H), Choice of ICC function (2, 3), Related Error (N, Y) and Number of Common Items (05, 15, 25). For example, H3Y05 refers to the cell with a high related error, 3PL function using 5 common items for linking.

Parameter and Data Generation

Non-related error conditions

The observed statistics of these item parameters were $\bar{a} = 1.2334$, $s_a = .3141$; $\bar{b} = .07216$, $s_b = 1.1868$; and $\bar{c} = .1227$, $s_c = .0669$ (see Table 6). All replications used these parameters to generate two unique sets of item parameter estimates (i.e., one for each form) as discussed in Chapter 3. Plots of the relationships between a and b as well as conditions are presented in Figures 1 and 2 (Appendix B contains the complete set of item parameters for both the non-related and related error conditions). These figures show that there is no discernible relationship between the a and b parameters ($r = .112$), or the c and b parameters ($r = .019$).

Figure 1
Non-related α parameters vs. non-related b parameters

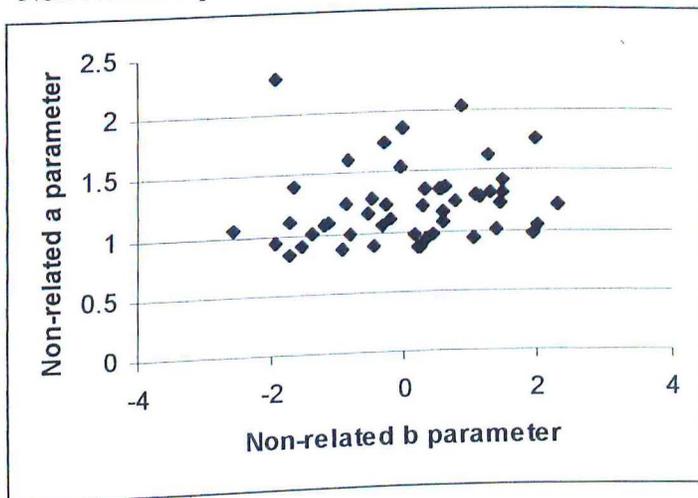
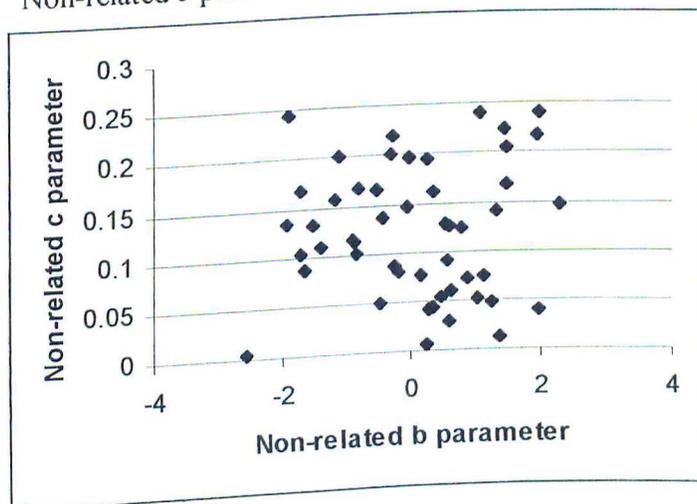


Figure 2
Non-related c parameters vs. non-related b parameters



Related error conditions

The observed statistics for the related error condition are as follows: $\bar{a} = 1.3450$, $s_a = .2963$; $\bar{b} = .0126$, $s_b = .843$; and $\bar{c} = .1262$, $s_c = .0369$ (see Table 6). These parameters exhibited the expected graphical relationships (with $r_{ab} = .08$ and $r_{cb} = .422$) (see Figures 3 and 4). Each replication for a related-error condition used these related

error parameters to generate two unique sets of item parameter estimates. The item parameter estimates had varying SEE's based upon the underlying b and the condition under study. Some examples of these relationships can be seen in Figures 5 and 6. For example, in the high related-error 3PL condition (Figure 5), the SEE of b exhibits a fishhook pattern when plotted against the b parameters. This is to be expected as the SE conditions identified in Table 3 show a very high level of SE for extreme negative values of b (i.e., 7.5) and a somewhat lower level of SE (i.e., 2.0) for extreme positive values of b . The other expected relationships among the related parameters and SE conditions were also found to exist throughout the study.

Table 6
Descriptive Statistics for Item Parameters

Parameter	N	Minimum	Maximum	Mean	Std. Dev.
Non-related a parameter	50	.8469	2.3059	1.2334	.3141
Non-related b parameter	50	-2.5556	2.2956	.0722	1.1868
Non-related c parameter	50	.0033	.2446	.1227	.0669
Related a parameter	50	.7754	2.0854	1.3450	.2963
Related b parameter	50	-2.5836	1.7188	.0126	.8430
Related c parameter	50	.0240	.2454	.1262	.0369

Figure 3 - Related a parameters vs. Related b parameters

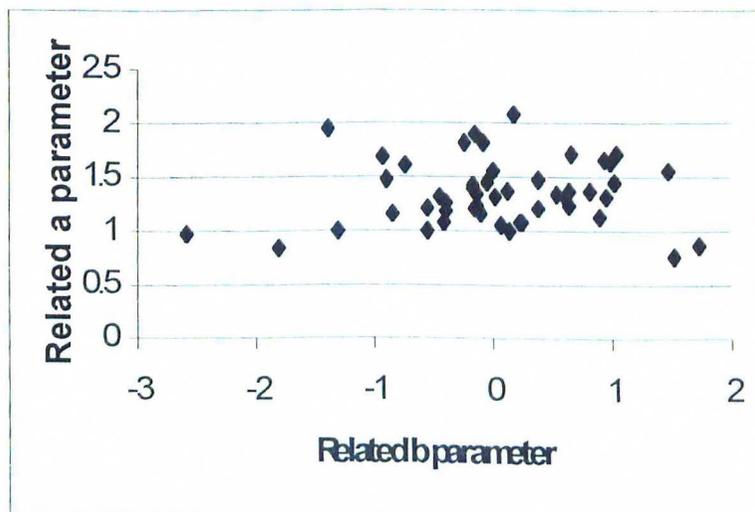


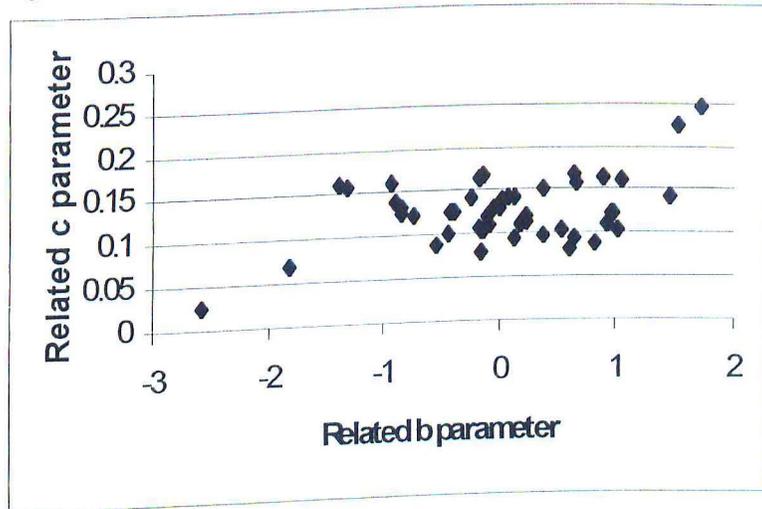
Figure 4 - Related c parameter vs. Related b parameter

Figure 5 - Example of H3Y05 Case

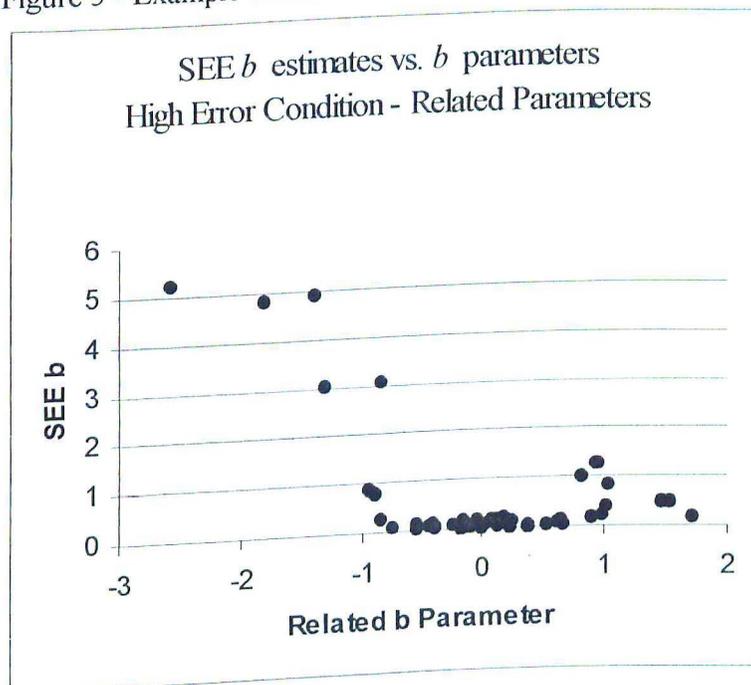
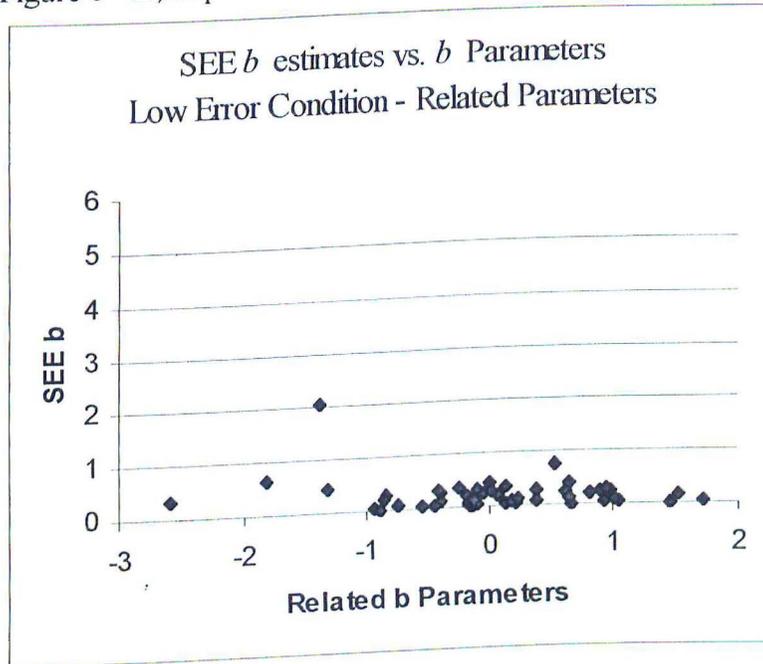
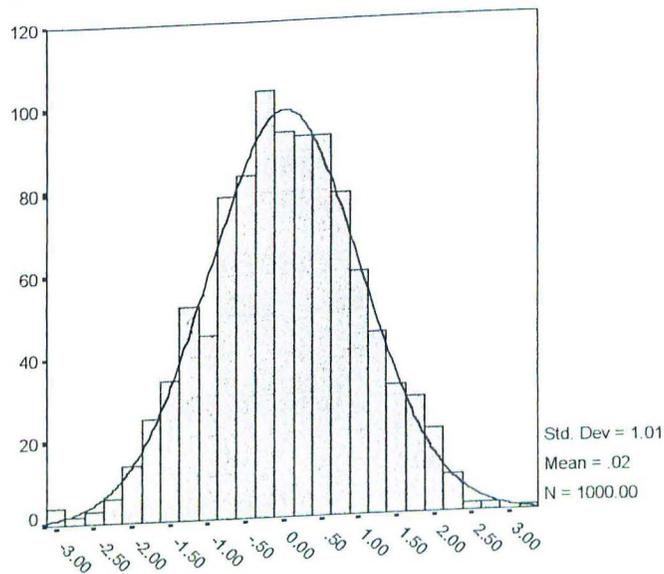


Figure 6 - Example of L3Y05 Case



All conditions. A set of 1000 random Z s was generated from a Normal[0,1] distribution. These Z s were considered to be θ s and the observed mean and standard deviation were .02 and 1.01, respectively (see Figure 7).

Figure 7 - Distribution of ability parameters used in study



Linking Study Results

Recovery of α and β parameters

Summary statistics for α and β are presented in Tables 7 and 8.

Non-related error conditions

For the non-related error cases, the mean value of α ranged from .9967 to 1.6212 across both 2PL and 3PL conditions (Table 7). It can be seen that the estimates for α and β more closely approached the parameter values (i.e., $\alpha = 1.0$, $\beta = 0.0$) as the amount of standard error was decreased. When combining the 2PL and 3PL cases, the mean $\hat{\alpha}$ s were 1.1703, 1.0547 and 1.0049 for the high, moderate and low error conditions, respectively. Better parameter estimation was also obtained by increasing the number of common items. By collapsing across error conditions and looking only at the number of common items, mean $\hat{\alpha}$ s were 1.1800, 1.0414 and 1.0085 for the 5, 15 and 25 common items respectively. In conjunction with these trends there was a corresponding decrease in the variability of the estimation of $\hat{\alpha}$ (i.e., variability was less if standard error was decreased or the number of common items was increased). Examples of these relationships are presented in Figures 8 and 9 (see Appendices C and D for complete set of figures).

Similar results were obtained for the β parameter estimation in the non-related conditions. Here the mean values of $\hat{\beta}$ ranged from -0.1458 to 0.2356 with a mean $\hat{\beta}$ of .0026 across all non-related cells. When combining all similar error conditions, the

mean $\hat{\beta}$ values were 0.0318 (high error condition), -0.0246 (moderate error condition) and 0.0006 (low error condition). When looking at the common items factor, the best mean estimation of β occurred for the 15 common items condition (with an overall mean of 0.0064) rather than with the 25 common items (with an overall mean of -0.0100). However, upon closer inspection of Table 7 it can be seen that better estimation of β was obtained with 25 items in all cases except the moderate and high SE 3PL cases.

Graphically, the differences between the $\hat{\beta}$ s for the 15 and 25 common items conditions appeared to be rather small (see Appendix D). The variability in the estimation of β also decreased with an increase in the number of common items across all conditions.

Related error conditions

The overall estimation of α and β was similarly good for the related-error conditions (Table 8). The mean $\hat{\alpha}$ ranged from 1.0001 to 1.5211 with a grand mean of 1.0748 across all related-error conditions. The estimation of α improved and approached its theoretical value of 1.0 from a mean $\hat{\alpha} = 1.1567$ in the high error conditions to a mean $\hat{\alpha} = 1.0198$ in the low error conditions ($\hat{\alpha} = 1.0479$ for the moderate error conditions).

The estimation of α also improved as the number of common items increased (from mean $\hat{\alpha} = 1.1590$ with 5 common items to mean $\hat{\alpha} = 1.0203$ with 25 common items). As in the non-related case, the variability of $\hat{\alpha}$ also decreased as the level of error was reduced or as the number of common items was increased.

The mean $\hat{\beta}$ ranged from -0.4313 to 0.0481 with a grand mean of -0.0490. As with the non-related case, the same pattern for β emerged in that these estimates more closely approached the parameter of $\beta = 0$ as the standard error level was reduced (with mean $\hat{\beta} = -0.0806$ at the high error conditions, mean $\hat{\beta} = -0.0505$ at the moderate error conditions, and mean $\hat{\beta} = -0.0160$ at the low error conditions). The estimation of β also was also better for the 15 and 25 common items conditions than for the 5 common items condition (see Appendix D) with a reduced variability of β estimation as the number of common items increased. In general, it may be noted that the variability of the α and β estimates is generally higher with increased levels of error.

Table 7

Means and Standard Deviations for α and β

Non-related error cases

2PL Model		Alpha				Beta			
		Min	Max	Mean	Std. Dev.	Min	Max	Mean	Std. Dev.
Error Level	Common								
Low	5	.91	1.10	1.0073	0.0346	-.17	.20	0.0159	0.0671
	15	.95	1.06	0.9967	0.0199	-.12	.10	-0.0053	0.0409
	25	.94	1.05	0.9997	0.0165	-.11	.11	-0.0028	0.0363
Moderate	5	.86	1.24	1.0221	0.0795	-.66	.68	-0.0001	0.1922
	15	.86	1.15	1.0053	0.0521	-.35	.26	0.0062	0.1145
	25	.90	1.10	0.9981	0.0370	-.19	.20	-0.0013	0.0711
High	5	.48	2.15	1.1742	0.2556	-1.93	1.63	-0.0141	0.5295
	15	.69	1.38	1.0334	0.1296	-.79	.66	0.0175	0.2571
	25	.74	1.39	0.9985	0.0937	-.47	.65	0.0136	0.1817
3PL Model									
Low	5	.78	1.50	1.0263	0.0980	-.63	.80	-0.0228	0.2371
	15	.83	1.15	0.9978	0.0570	-.39	.37	0.0261	0.1449
	25	.87	1.14	1.0018	0.0452	-.31	.25	-0.0074	0.0994
Moderate	5	.72	2.41	1.2289	0.2783	-2.51	1.89	-0.1458	0.7032
	15	.78	1.53	1.0608	0.1231	-1.17	1.07	0.0122	0.3697
	25	.79	1.35	1.0129	0.0956	-.75	.73	-0.0189	0.2661
High	5	.33	5.67	1.6212	0.8329	-9.54	9.52	0.2356	2.6492
	15	.56	2.18	1.1545	0.2992	-3.78	2.80	-0.0183	1.1065
	25	.58	1.94	1.0399	0.2047	-2.23	2.21	-0.0434	0.7969

Table 8
Means and Standard Deviations for α and β

Related error cases

2PL Model		Alpha				Beta			
		Min	Max	Mean	Std. Dev.	Min	Max	Mean	Std. Dev.
Error Level	Common								
Low	5	.87	1.19	1.0001	0.0588	-.21	.17	0.0005	0.0717
	15	.91	1.10	1.0058	0.0304	-.08	.09	0.0097	0.0328
	25	.93	1.07	1.0027	0.0240	-.05	.06	0.0038	0.0225
Moderate	5	.75	1.43	1.0117	0.1081	-.47	.41	-0.0157	0.1454
	15	.85	1.17	1.0116	0.0553	-.19	.18	0.0001	0.0656
	25	.88	1.10	1.0039	0.0412	-.14	.14	-0.0035	0.0477
High	5	.24	1.98	1.1249	0.2508	-1.40	1.79	0.0082	0.3938
	15	.64	1.47	1.0359	0.1405	-.62	.56	0.0085	0.1769
	25	.73	1.33	1.0040	0.1042	-.32	.43	-0.0061	0.1333
3PL Model									
Low	5	.65	1.57	1.0837	0.1835	-1.23	.88	-0.1007	0.3196
	15	.80	1.27	1.0156	0.0845	-.31	.37	0.0012	0.1296
	25	.85	1.17	1.0108	0.0608	-.27	.21	-0.0104	0.0971
Moderate	5	.44	2.75	1.2122	0.3839	-2.60	1.19	-0.2857	0.6542
	15	.59	1.55	1.0354	0.1654	-.84	.89	0.0328	0.2940
	25	.67	1.37	1.0128	0.1159	-.67	.58	-0.0313	0.2091
High	5	.00	9.40	1.5211	1.1363	-10.15	4.87	-0.4313	1.6055
	15	.05	4.22	1.1668	0.5373	-2.71	1.59	0.0481	0.5401
	25	.26	2.39	1.0877	0.3609	-1.73	.82	-0.1109	0.3726

Figure 8 - Mean alphas for non-related 2PL conditions

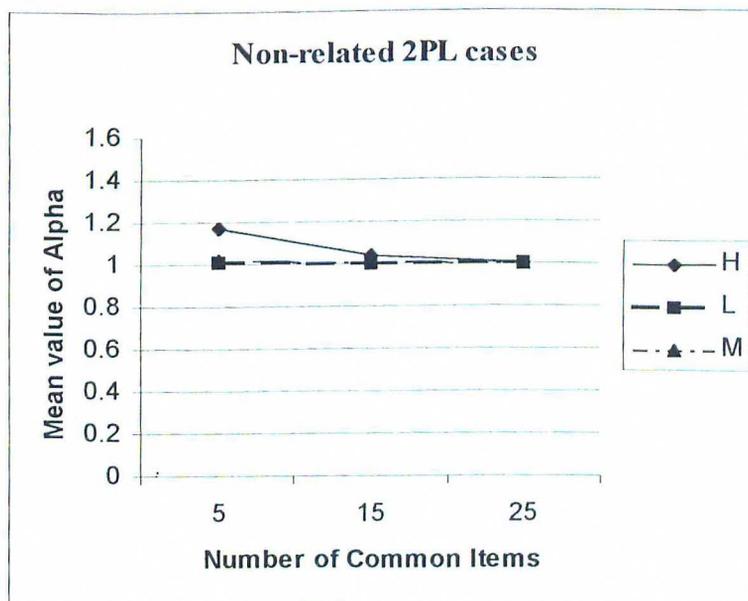
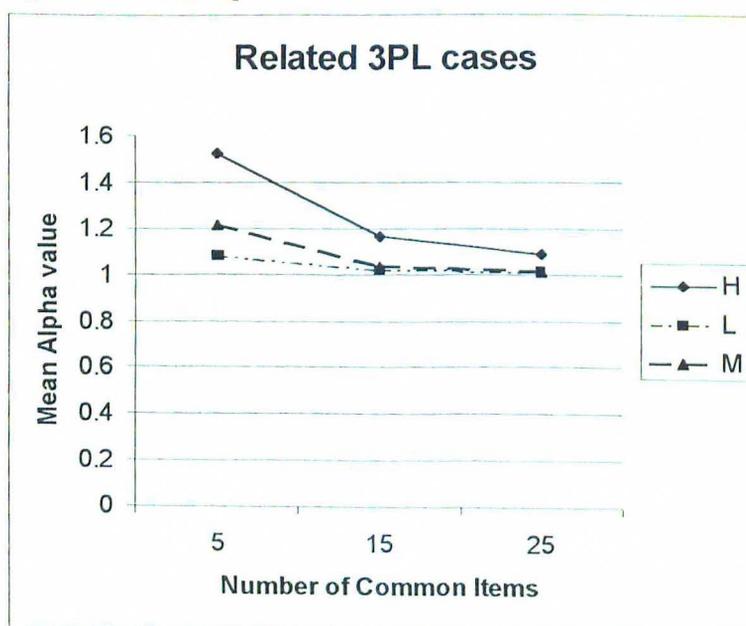


Figure 9 - Mean alphas for related 3PL conditions



RMSE and Bias in α and β estimation

The overall accuracy of α and β estimation was also reflected in their RMSE and bias values. The values obtained here correspond with the mean values of $\hat{\alpha}$ and $\hat{\beta}$ discussed earlier. The RMSE and Bias terms for α and β for all cells can be found in Tables 9 and 10 (Appendices E and F contain figures depicting the relationship between the RMSE and Bias as a function of the number of common items and error level).

Non-related error conditions

The overall RMSE(α) was 0.1755 across all non-related error test conditions (with a range of 0.0165 to 1.0378). These RMSE values improved as the level of error was reduced (with mean RMSE(α) = 0.3530 for the high error conditions, mean RMSE(α) = 0.1275 for the moderate error conditions and mean RMSE(α) = 0.0459 for the low error conditions). The RMSE(α) was also affected by the number of common items with 5 common items performing worst (mean RMSE(α) = 0.3209 across all non-related error test conditions) and 25 common items performing best (mean RMSE(α) = 0.0828 across all non-related error test conditions). The Bias(α) for the non-related error conditions was 0.0766 across all 18 non-related error cells. The mean Bias(α) values followed the same trends as the RMSE(α) values both as the level of error was reduced (mean Bias(α) = 0.1703 for high error conditions, mean Bias(α) = 0.0547 for moderate error conditions, mean Bias(α) = 0.0049 for low error conditions) and the number of common items was increased (mean Bias(α) = 0.1800 for 5 common items conditions, mean Bias(α) = 0.0414 for 15 common items conditions and mean Bias(α) = .0085 for 25 common items

conditions).

The overall mean $RMSE(\beta)$ term in the non-related error conditions was somewhat higher at 0.4379 ranging from a low of 0.0364 to a high of 2.6544. The $RMSE(\beta)$ values followed the same trend as the $RMSE(\alpha)$ values in that the mean $RMSE(\beta)$ values decreased as the error level decreased (mean $RMSE(\beta) = 0.9205$ for high error conditions, mean $RMSE(\beta) = 0.2882$ for moderate error conditions, mean $RMSE(\beta) = 0.1051$ for low error conditions) and as the number of common items increased (mean $RMSE(\beta) = 0.7330$ for 5 common items condition, mean $RMSE(\beta) = 0.3389$ for 15 common items condition and mean $RMSE(\beta) = .2419$ for 25 common items condition). The $Bias(\beta)$ also lessened as the error level condition was lowered (mean $Bias(\beta) = 0.0318$ for high error conditions, mean $Bias(\beta) = -0.0246$ for moderate error conditions and mean $Bias(\beta) = 0.0006$ for low error conditions). The mean $Bias(\beta)$ term decreased from the 5 common items condition (0.0115) to the 15 common items condition (0.0064) but increased again at the 25 common items condition (-0.0100). This is likely due to the large values for the high error, 3PL 5 and 15 item conditions (Table 9).

Related error conditions

Although the Bias and RMSE values for α and β under the related-error conditions were somewhat higher than for the non-related conditions, they do correspond with the recovery of the α and β parameters. The $RMSE(\alpha)$ ranged from 0.0241 to 1.2480 with a mean of 0.2290 across all related error conditions. The related error conditions showed the same trends as the non-related error conditions with smaller mean $RMSE(\alpha)$ s associated with lower error conditions (mean $RMSE(\alpha) = 0.4515$ for high

error conditions, mean $RMSE(\alpha) = 0.1585$ for moderate error conditions and mean $RMSE(\alpha) = 0.0771$ for low error conditions). Also, smaller mean $RMSE(\alpha)$ s were associated with an increase in the number of common items (mean $RMSE(\alpha) = 0.3928$ for the 5 common items conditions, mean $RMSE(\alpha) = 0.1745$ for the 15 common items conditions and mean $RMSE(\alpha) = 0.1197$ for the 25 common items conditions). The $Bias(\alpha)$ ranged from 0.0001 to 0.5211 with a mean of 0.0748 across all related error conditions. Obtained values for the mean $Bias(\alpha)$ terms were 0.1567 (high error conditions), 0.0479 (moderate error conditions) and 0.0198 (low error conditions). When the common items factor was analyzed, mean $Bias(\alpha)$ terms were 0.1590 (5 common items), 0.0452 (15 common items) and 0.0203 (25 common items).

The $RMSE(\beta)$ term ranged from 0.0228 to 1.659 with an average $RMSE(\beta)$ of 0.3033 across all related error conditions. The same pattern which was observed in the non-related error conditions was observed under these conditions as well, with the mean $RMSE(\beta)$ decreasing as the level of error was reduced (mean $RMSE(\beta) = 0.5486$ for the high error conditions, mean $RMSE(\beta) = 0.2463$ for the moderate error conditions, mean $RMSE(\beta) = 0.1150$ for the low error conditions) and as number of common items increased (mean $RMSE(\beta) = 0.5528$ for the 5 common items conditions, mean $RMSE(\beta) = 0.2070$ for the 15 common items conditions and mean $RMSE(\beta) = 0.1500$ for the 25 common items conditions). Again, the $Bias(\beta)$ under these conditions was the same as the mean $\hat{\beta}$ s previously discussed.

Table 9

Bias and RMSE for α and β Estimates

Non-Related Error Cases

Model	Error Level	Common	Alpha		Beta	
			Bias	RMSE	Bias	RMSE
2PL	Low	5	0.0073	0.0353	0.0159	0.0688
		15	-0.0033	0.0201	-0.0053	0.0412
		25	-0.0003	0.0165	-0.0028	0.0364
	Moderate	5	0.0221	0.0824	-0.0001	0.1918
		15	0.0053	0.0523	0.0062	0.1144
		25	-0.0019	0.0370	-0.0013	0.0710
	High	5	0.1742	0.3089	-0.0141	0.5287
		15	0.0334	0.1336	0.0175	0.2572
		25	-0.0015	0.0935	0.0136	0.1819
3PL	Low	5	0.0263	0.1013	-0.0228	0.2378
		15	-0.0022	0.0569	0.0261	0.1469
		25	0.0018	0.0451	-0.0074	0.0995
	Moderate	5	0.2289	0.3599	-0.1458	0.7168
		15	0.0608	0.1371	0.0122	0.3692
		25	0.0129	0.0963	-0.0189	0.2662
	High	5	0.6212	1.0378	0.2356	2.6544
		15	0.1545	0.3362	-0.0183	1.1045
		25	0.0399	0.2081	-0.0434	0.7965

Table 10

Bias and RMSE for α and β Estimates

Related Error Cases

Model	Error Level	Common	Alpha		Beta	
			Bias	RMSE	Bias	RMSE
2PL	Low	5	0.0001	0.0587	0.0005	0.0715
		15	0.0058	0.0308	0.0097	0.0341
		25	0.0027	0.0241	0.0038	0.0228
	Moderate	5	0.0117	0.1085	-0.0157	0.1460
		15	0.0116	0.0564	0.0001	0.0655
		25	0.0039	0.0413	-0.0035	0.0477
	High	5	0.1249	0.2797	0.0082	0.3931
		15	0.0359	0.1448	0.0085	0.1767
		25	0.0040	0.1041	-0.0061	0.1331
3PL	Low	5	0.0837	0.2014	-0.1007	0.3345
		15	0.0156	0.0858	0.0012	0.1293
		25	0.0108	0.0616	-0.0104	0.0975
	Moderate	5	0.2122	0.4607	-0.2857	0.7127
		15	0.0354	0.1678	0.0328	0.2952
		25	0.0128	0.1164	-0.0313	0.2111
	High	5	0.5211	1.2480	-0.4313	1.6593
		15	0.1668	0.5616	0.0481	0.5412
		25	0.0877	0.3707	-0.1109	0.3881

F_{\max} , F_{converge} and IR Results

The average values for F_{\max} , F_{converge} and IR were computed for all replications and are reported in Tables 11, 12 and 13 (Appendices G, H and I contain all corresponding figures).

Non-related conditions

The mean values for F_{\max} ranged from .0036 to 3.0273 for the non-related error conditions with a grand mean of 0.4574. The mean values for F_{converge} ranged from 0.0007 to 0.1894 for the non-related error conditions with a grand mean of 0.0341. The values for both F_{\max} and F_{converge} decreased as the amount of error condition decreased. Mean F_{\max} values for all cases within specified error conditions were 1.0706 (high error conditions), 0.2544 (moderate error conditions) and 0.0472 (low error conditions). These values can be compared to the mean F_{converge} values of 0.0738 (high error conditions), 0.0231 (moderate error conditions) and 0.0054 (low error conditions).

The improvement ratio was introduced to determine the improvement gained by conducting the linking where a higher IR indicates a better linking between two forms. For this set of conditions, mean IR ranged from 0.7418 to 0.8727 with a grand mean of 0.8087 (Table 13). The IR also decreased with a decrease in the error level condition. Mean IRs for all high error conditions was 0.8287 while it was 0.8106 and 0.7868 for the moderate and low error conditions, respectively. When IR was looked at for all cases within the separate common items conditions, the mean IR values 0.8081 (the 5 common items condition), 0.8159 (the 15 common items condition) and 0.8021 (the 25 common items condition) were found. When the 2PL cases were compared against the 3PL cases,

it could also be seen that higher IRs were obtained for the 3PL cases (which had more error in the item parameter estimates) for otherwise comparable cells (e.g., 2PL, Low error, 5 common items mean IR = 0.7418 whereas 3PL, Low error, 5 common items mean IR = 0.8111).

Related error conditions

The mean values for F_{\max} in the related error cases ranged from 0.0059 to 2.2037, with a grand mean of 0.3476 for the related error conditions. As in the non-related conditions, mean F_{\max} values decreased as the error level decreased with values of 0.7542, 0.2274 and 0.0612 for the high error, moderate error and low error conditions, respectively. The mean values of F_{converge} ranged from 0.0013 to 0.4206 with a mean value of 0.0640 across all related-error conditions. Although this mean F_{converge} value obtained under the related-error conditions appears almost twice as large as the one obtained under the non-related error conditions, this can be attributed almost entirely to the very high error conditions associated with the related error 3PL simulations (see Table 12). The F_{converge} values followed the same trend as the F_{\max} values with mean F_{converge} being 0.1461, 0.0350 and 0.0109 for the high error, moderate error and low error conditions, respectively.

For this set of conditions, mean IR ranged from 0.6941 to 0.7835 with a grand mean of 0.7446 across all related error conditions (Table 13). Once more, the mean IRs decreased as the error level decreased, with the values 0.7566, 0.7446 and 0.7327 being obtained for the high error, moderate error and low error conditions, respectively. Mean IRs were also obtained for all related cases combined by number of common items. These

values were 0.7353, 0.7394 and 0.7592 for the 5 common item, 15 common item and 25 common item conditions, respectively.

All conditions

It can be seen that the mean values for F_{\max} and F_{converge} increased with an increase in the error associated with the item parameter estimates. This is possibly explained in that more error might make the TCCs more dissimilar from each other which would affect both F_{\max} and F_{converge} (i.e., there could be more discrepancy to account for between the forms).

The Improvement Ratio showed a trend of larger ratios being associated with higher error conditions. It can be seen that for both the non-related error and related error conditions, IR was generally larger as the level of SE was increased. The same trend was found comparing the 3PL conditions (i.e., more error) to the 2PL conditions (i.e., less error).

Table 11

Means and Standard Deviations for F_{\max} and F_{converge}

Non-related error cases

Model	Error Level	Common	F_{\max}		F_{converge}	
			Mean	Std. Dev.	Mean	Std. Dev.
2PL	Low	5	0.0036	0.0030	0.0007	0.0007
		15	0.0110	0.0099	0.0015	0.0014
		25	0.0227	0.0235	0.0023	0.0020
	Moderate	5	0.0219	0.0222	0.0029	0.0028
		15	0.0764	0.0700	0.0075	0.0067
		25	0.0948	0.0852	0.0117	0.0122
	High	5	0.1392	0.1223	0.0157	0.0178
		15	0.3641	0.3145	0.0392	0.0352
		25	0.5631	0.5125	0.0627	0.0574
3PL	Low	5	0.0255	0.0268	0.0025	0.0024
		15	0.0906	0.0882	0.0089	0.0086
		25	0.1298	0.1403	0.0165	0.0177
	Moderate	5	0.1541	0.1479	0.0121	0.0137
		15	0.4551	0.4615	0.0418	0.0485
		25	0.7240	0.7771	0.0624	0.0557
	High	5	0.5154	0.5436	0.0276	0.0396
		15	1.8144	1.8500	0.1084	0.1255
		25	3.0273	3.5145	0.1894	0.1967

Table 12

Means and Standard Deviations for F_{\max} and F_{converge}

Related error cases

Model	Error Level	Common	F_{\max}		F_{converge}	
			Mean	Std. Dev.	Mean	Std. Dev.
2PL	Low	5	0.0059	0.0049	0.0013	0.0011
		15	0.0131	0.0121	0.0024	0.0022
		25	0.0183	0.0149	0.0024	0.0021
	Moderate	5	0.0211	0.0185	0.0042	0.0042
		15	0.0433	0.0369	0.0072	0.0075
		25	0.0668	0.0600	0.0102	0.0099
	High	5	0.1081	0.1273	0.0184	0.0244
		15	0.2684	0.2701	0.0425	0.0547
		25	0.4442	0.4126	0.0606	0.0641
3PL	Low	5	0.0566	0.0639	0.0081	0.0094
		15	0.1076	0.1090	0.0220	0.0259
		25	0.1659	0.1635	0.0294	0.0329
	Moderate	5	0.1774	0.2009	0.0245	0.0252
		15	0.4275	0.4827	0.0670	0.0740
		25	0.6285	0.7358	0.0969	0.1134
	High	5	0.4200	0.4523	0.0709	0.1034
		15	1.2609	1.2206	0.2637	0.3519
		25	2.0237	2.0268	0.4206	0.4506

Table 13

Improvement Ratios Means and Deviations

Model	Error Level	Common	Improvement Ratio (IR)			
			Non-Related Error		Related Error	
			Mean	Std. Dev.	Mean	Std. Dev.
2PL	Low	5	0.7418	0.2256	0.6941	0.2502
		15	0.7845	0.2127	0.7010	0.2607
		25	0.7823	0.2333	0.7768	0.2192
	Moderate	5	0.7762	0.2325	0.7090	0.2592
		15	0.8144	0.2143	0.7560	0.2346
		25	0.7946	0.2131	0.7458	0.2381
	High	5	0.8044	0.2257	0.7425	0.2467
		15	0.8034	0.2085	0.7677	0.2247
		25	0.7930	0.2236	0.7835	0.2106
3PL	Low	5	0.8111	0.2056	0.7568	0.2295
		15	0.8174	0.2056	0.7194	0.2406
		25	0.7835	0.2104	0.7479	0.2248
	Moderate	5	0.8423	0.2002	0.7569	0.2297
		15	0.8216	0.2009	0.7378	0.2466
		25	0.8143	0.2166	0.7622	0.2226
	High	5	0.8727	0.1853	0.7524	0.2390
		15	0.8539	0.2003	0.7543	0.1856
		25	0.8447	0.2360	0.7391	0.1899

Equating study results

This phase of the study examined the robustness of an equating by computing average RMSE for θ and true scores. The results of this analysis are presented in Tables 14 and 15 (for graphical analyses see Appendices J and K).

Non-related error conditions

Mean RMSE(θ) for the 2PL cases ranged from .5085 to .7904, while for the 3PL cases they ranged from .5567 to 3.0426 (Table 14). The $\hat{\theta}$'s generally appeared to be better for lower error conditions and more common items.

When the RMSE(ξ)s were analyzed, a great deal of agreement was seen across the error levels for the 2PL non-related error cases. The mean RMSE(ξ) values ranged from 4.5273 to 4.6410. There was a noticeable increase in the variability of these estimates, with more variability being associated with the high error condition (Table 14). This is not unexpected in that the higher error condition did have a somewhat poorer linking, as previously discussed. However, it is interesting to note that the mean values for the RMSE(ξ)s were relatively constant across the error levels, suggesting that the equating was not unduly affected by the higher error cases.

The 3PL cases had mean RMSE(θ)s which ranged from .5567 to 3.0426 and mean RMSE(ξ)s ranging from 3.5260 to 4.1988. Again, an increase in the variability of the estimates was seen with an increase in the level of error in the item parameter estimates. In these 3PL cases there was slightly more variability in the mean RMSE(ξ)s, yet all mean errors were still within a point of each other, again suggesting that the equated scores

were fairly robust across these test conditions. It is also interesting to note that overall the RMSE(ξ)s appear somewhat smaller in the 3PL cases compared to the 2PL cases. A true comparison cannot be made across these levels however, in that different response data were used to compute the $\hat{\theta}$ s and $\hat{\xi}$ s, which could affect these results.

Related error conditions

The related error conditions showed very similar results to the non-related error conditions. For the 2PL cases, the mean RMSE(θ) ranged from .4450 to .6001 while the mean RMSE(ξ) ranged from 4.5601 to 4.6264 (Table 15). Again, while there was slightly more error associated with the $\hat{\xi}$ s obtained in the high error condition, the difference in the obtained $\hat{\xi}$ error across the 2PL models was almost negligible.

The 3PL cases had mean RMSE(θ) values which ranged from .3482 to 1.5172 and mean RMSE(ξ)s which ranged from 2.8566 to 3.9853. While these cases exhibited more variability in the mean RMSE(ξ)s, the absolute difference in the estimates was just slightly over 1 point (1.1287) suggesting robustness of the estimates across the error conditions.

Table 14
 RMSE of θ and True Scores between converted and parameters
 Means and Standard Deviations
 Non-related error cases

2PL Model		θ RMSE		True Score RMSE	
		Mean	Std. Dev.	Mean	Std. Dev.
Error Level	Common				
Low	5	.5300	.0490	4.6299	.0782
	15	.5085	.0303	4.6319	.0765
	25	.5123	.0253	4.6251	.0757
Moderate	5	.5523	.1361	4.6410	.1739
	15	.5289	.0735	4.6112	.1683
	25	.5163	.0504	4.6214	.1685
High	5	.7904	.3775	4.5431	.3707
	15	.5982	.1676	4.5273	.3470
	25	.5552	.1219	4.5510	.3516
3PL Model					
Low	5	.5567	.1276	3.5291	.1583
	15	.4895	.0680	3.5260	.1632
	25	.4868	.0553	3.5297	.1504
Moderate	5	1.0502	.5019	3.5891	.3365
	15	.6867	.1924	3.5393	.3249
	25	.6174	.1368	3.5610	.3241
High	5	3.0426	2.1487	4.1988	.7562
	15	1.6943	.7029	4.1915	.8084
	25	1.3750	.5227	4.0455	.6803

Table 15
 RMSE of θ and True Scores between converted and parameters
 Means and Standard Deviations
 Related error cases

2PL Model		θ RMSE		True Score RMSE	
Error Level	Common	Mean	Std. Dev.	Mean	Std. Dev.
Low	5	.4450	.0420	4.5630	.0268
	15	.4466	.0181	4.5601	.0256
	25	.4420	.0117	4.5604	.0263
Moderate	5	.4548	.0777	4.5694	.0554
	15	.4474	.0375	4.5639	.0583
	25	.4408	.0256	4.5630	.0576
High	5	.6001	.2332	4.6191	.1622
	15	.4883	.1060	4.6264	.1666
	25	.4565	.0674	4.6127	.1642
3PL Model					
Low	5	.4702	.1630	2.8591	.0211
	15	.3482	.0309	2.8566	.0202
	25	.3335	.0178	2.8580	.0207
Moderate	5	.7871	.4653	3.0114	.0769
	15	.4572	.1171	3.0105	.0745
	25	.4040	.0612	3.0102	.0715
High	5	1.5172	1.3066	3.9411	.5714
	15	.7505	.3686	3.9555	.4438
	25	.6047	.2225	3.9853	.4391

Chapter V - Discussion

Linking Study

Recovery of α and β parameters

The approach used in this study was to directly sample item parameter estimates from known sampling distributions, rather than obtaining estimates by calibration. The approach used permitted direct control of the degree of error in the item parameter estimates. The forms being equated used identical item parameters. The differences between the item parameter estimates for any pair of equated forms were due only to the error of the sampling distribution from which they were obtained. In this case, a perfect linking would put the initial and target scales onto a common metric and reproduce the parameters $\alpha = 1$ and $\beta = 0$. The closer the linking parameter estimates were to these values, the more closely the TCCs could be said to be matched.

Non-related error conditions

As expected, the worst estimation of α and β occurred for the higher error conditions. This was true both within an ICC function type (e.g., high error conditions for the 2PL cases vs. low error conditions for the 2PL cases) and for similar error levels across function type (e.g., low error conditions for the 2PL cases vs. low error conditions for the 3PL cases). This effect was especially noticeable for the 5 common items conditions. However, once 15 or 25 common items were used the estimates of α more closely approached the parameter, even in the case of high error. This result is in agreement with Baker (1996), Kolen (1990) and Wingersky and Lord (1984) all of which suggested using a minimum of 15 common items for equating. Even in the higher error

conditions, the use of 15 common items showed much better recovery of the α and β parameters. The worst estimation of α occurred for the high error 3PL condition which had an $SE(b) = 2.73$ (see Table 2). In this condition, the mean of the $\hat{\alpha}$ s dropped from 1.6212 with 5 common items to 1.1545 with 15 common items. This pronounced improvement of the mean $\hat{\alpha}$ s when using 15 or 25 common items at higher error levels is suggestive of an interaction between the amount of error in the item parameter estimates and the number of common items used to link two forms (see Appendix C) (i.e., as more error is introduced into the item parameter estimates, it becomes more important to have at least 15 common items for the linking). The mean $\hat{\beta}$ s also showed closer agreement with the β parameter (i.e., 0) as the error was reduced.

It can generally be observed that in the 2PL non-related cases the $\hat{\alpha}$ s and $\hat{\beta}$ s more closely matched the α and β parameters, with less variation, than in the 3PL non-related cases. This would be expected in that the 2PL model had less error in the item parameter estimates, and therefore the forms were more likely to be similar initially and therefore more easily linked.

Related-error conditions

These results were very similar to the non-related conditions. It was seen that the estimation of α and β was affected by the amount of error in the item parameter estimates (i.e., the modeled error level and the use of a 3PL function over a 2PL function). As the level of SE was increased, the estimates of α and β became less accurate with an increase

in variance across the replications. This inaccuracy of estimation was magnified when only 5 common items were used (again suggestive of an interaction between error level and number of common items used for the linking).

Bias and RMSE of α and β estimation

Non-related error conditions

As the amount of error in item parameter estimation is increased, the RMSEs of the equating coefficients increased. Both $RMSE(\alpha)$ and $RMSE(\beta)$ were smaller for the lower error conditions than for the higher error conditions (i.e., low level of error conditions produced the smallest error while high level of error conditions produced the largest RMSEs). This was especially noticeable when only 5 common items were used in the equating. As expected, the $RMSE(\alpha)$ and $RMSE(\beta)$ calculations were better for the 15 and 25 common items conditions. This can probably be attributed to greater stability in the estimation of α and β with more items (although the mean β estimate appeared somewhat worse for the 25 common item than for the 15 common item level in several cases, the variance of the estimates was consistently smaller with an increase in the number of common items reducing the $RMSE(\beta)$ value). The $Bias(\alpha)$ estimates showed the same trend as the $RMSE(\alpha)$ estimates in that Bias was greater with the higher error conditions and fewer common items. The same type of interaction noted before appeared in the values of the $Bias(\alpha)$ terms. Because in this study $\beta = 0$, the $Bias(\beta)$ estimates were the same as the mean $\hat{\beta}$ s.

Related-error conditions

As in the discussion of the recovery of the α and β parameters, the Bias and RMSE terms for α and β were affected by an increase in the amount of error in the item parameter estimates. This trend was seen both as the level of standard error was increased (i.e., from low to moderate to high) and as the ICC function was changed from the 2PL model to the 3PL model. Also, as in the non-related error conditions, the RMSEs and Bias terms for α and β generally improved with the addition of more common items.

F_{\max} , F_{converge} and IR Results

Tables 11 and 12 showed that the values of F_{\max} and F_{converge} actually increased with an increase in the number of common items (also see Appendices H and I). These larger F values associated with more common items appear to be a function of how F_{\max} and F_{converge} were calculated. Namely, when fewer items were used to link the two forms there was less opportunity for the modeled TCCs to differ by larger amounts thereby giving a smaller F value (Baker, 1996). Results similar to this (i.e., larger F values with more common items) were found by Baker.

Again, these findings suggest that overall the linking was successful, even in the higher error conditions. Not only were F_{converge} values relatively small throughout the conditions, the IR values improved with an increase in error (both in the level of SE modeled and switching from a 2PL function to a 3PL function). This is reflective of the fact that the higher error conditions had larger F_{\max} values while still obtaining relatively small F_{converge} values. Therefore, the IR would be correspondingly larger. The mean IR

increased as the error in the study increased with the high error conditions having the largest IR values for a particular parameter and model type. It was also seen that IR was greater for the 3PL models than for the corresponding 2PL models.

Equating Study

Another component to this study was to look at the effect of the linking obtained in the first part on true score equating to see if this was influenced by the errors in the item parameter estimates. Whereas the linking study looked at the accuracy of the linking process, this study was designed to assess whether there were any practical effects of the linking on the equated ability estimates. While θ estimation varied somewhat, each of the four groupings of results for the equating study (non-related 2PL, non-related 3PL, related 2PL and related 3PL cases) showed a great deal of similarities in their estimation of true scores.

As would be expected, the estimation of θ was better when the linking more accurately recovered the equating coefficients (i.e., better estimation occurred with lower error and more common items). The variability of the estimation also followed the same trends. Perhaps more interesting is that there was little variation in the estimation of ξ for a particular grouping, even when the $\hat{\theta}'$ estimation varied across the conditions within the grouping. For example, in the non-related 3PL cases, mean RMSE(θ) estimates ranged from .5567 to 3.0426 while mean RMSE(ξ) ranged from 3.5260 to 4.1988. The largest range for mean RMSE(ξ) was in the related 3PL conditions (these conditions had the most error attached to the item parameter estimates), however the maximum difference between

the mean $RMSE(\xi)$ values was only 1.1287 points. Considering that the $\hat{\xi}$ s were based on a 50 point test, this absolute difference still suggests a fairly robust equating (i.e., approximately a 2% difference).

General Discussion

The results clearly suggest that 5 common items can lead to a less adequate solution than 15 or 25 items. The values of $\hat{\alpha}$ and $\hat{\beta}$ do vary somewhat as a function of the level of error. This suggests that the increased error in the item parameter estimates leads the Davidson-Fletcher-Powell minimization algorithm to model what appears to be a vertical equating rather than a horizontal equating. When there was larger error in the item parameter estimates, the values obtained for α and β reflected the differences between the forms (as it did for all linkings). Therefore, as the evidence suggests, higher error in the item parameter estimates results in TCCs which are further apart and the estimation of α and β will adjust for this accordingly.

Also, it was seen that the F_{converge} values were generally very good for the linkings and that large initial differences between the TCCs being equated did not necessarily hinder this convergence. This was reflected in the values of IR which showed successive improvement as the error condition increased for conditions with common parameters. The same pattern emerged for both the non-related error parameters and the related error parameters conditions.

This study also showed that the error in the $\hat{\xi}$ s was relatively constant for a given

ICC function (i.e., 2PL or 3PL) and item parameters (i.e., non-related error and related error). This finding would indicate that even when there is a fair amount of error in the item parameter estimates, the equating itself is relatively robust.

Implications for future research

One limitation of this study is that it is difficult to make direct comparisons across the non-related error and related error conditions because two different sets of parameters were used. As was seen in the item parameter distribution, the non-related error parameters encompassed a wider range of b than did the related error parameters. Unfortunately, this was due to the random nature of the selection of the item parameters. It is possible that more extreme values of b , especially in the related error cases could produce more error in the item parameter estimates which in turn could cause the equating of the forms to perform more poorly. On the other hand, it is entirely likely that the item parameter distributions obtained in this study reflect a realistic situation. A possible enhancement to this study would be to use an empirical set of item parameter estimates. The item parameter estimates obtained from this data set could then be treated as the item parameter values, and the sampling distributions used in this study to obtain item parameter estimates could be adjusted accordingly (i.e., the data set estimates would provide the mean value and SEEs of the sampling distributions used in this study). By using an existing data set, any relationships among the parameters would naturally be modeled as well.

This study could be further extended by modifying the length of the test used for the equating and true score estimation. It was seen that 15 and 25 common items

provided good equating coefficient recovery in most circumstances. Yet these numbers represent 30% and 50% of the total items on the forms. It is possible that if the forms were longer the TCCs which are equated might not be as representative of the entire form, causing a worse case of true score equating. It would be interesting to examine whether the TCC equating remains robust when the number of common items is reduced as a percentage of the entire form length. Similarly, the common items could also be treated as an external anchor and the affect on true score equating could be determined.

This study was concerned only with horizontal equating. Another extension to this study would be to model a vertical equating situation. This study showed that for high error conditions, the equating coefficients were similar to what would be expected for a vertical equating. As discussed, this was because the larger differences between the two forms made the forms appear as if one was more difficult than the other. If the forms actually did differ on their level of difficulty in the item parameters, then any error in the estimation of the item parameters could impact the equating even more. For example, it could be entirely possible for two forms of different difficulty to be treated as two forms of equal difficulty (i.e., horizontal forms) which could cause error both in the equating coefficients and equated true scores.

Appendix A

References for design decisions

Study Factor	Reference(s)
Use of TCC approach	Baker & Al-Karni (1991); Hambleton & Swaminathan (1985); Kolen & Brennan (1995); Stocking & Lord (1983)
Use of 2PL and 3PL functions	Kolen & Whitney (1982); Petersen et al., (1983)
Amount of Standard Errors	Thissen & Wainer (1982)
Internal anchor test	Cook & Eignor (1991)
Use of EQUATE program	Baker (1992, 1993, 1996); Baker, Al-Karni, & Al-Dosary (1991);
Use of 5, 15, and 25 common items	Baker (1996); Kolen (1990); Wingersky & Lord (1984)
Relationship of SE with b	Thissen & Wainer (1982)
item parameter distributions	Harwell & Janosky (1991); Skaggs & Lissitz (1988); Thissen & Wainer (1982)
EAP estimation	Bock & Mislevy (1982); DeAyala, Schafer, & Sava-Bolesta (1995)

Appendix B

Item parameter distributions

Figure B1 - non-related α parameters

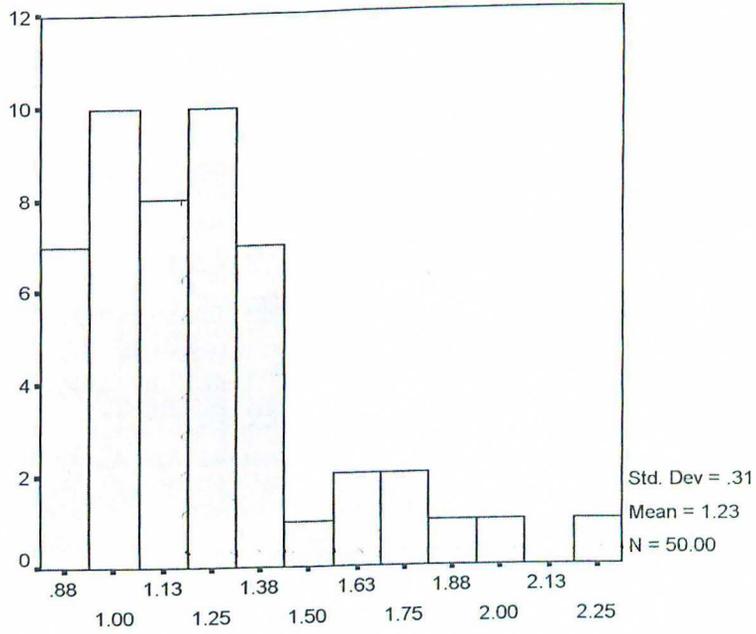


Figure B2 - Non-related b parameters

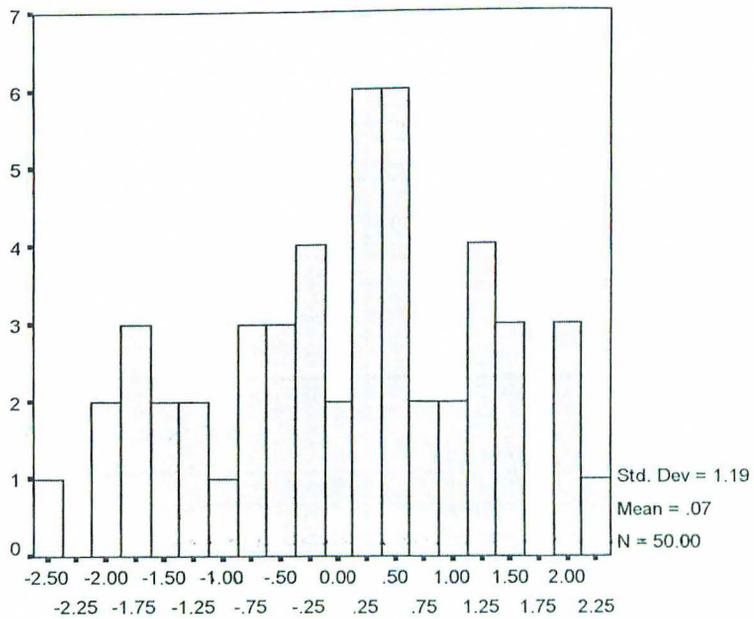


Figure B3 - Non-related c parameters

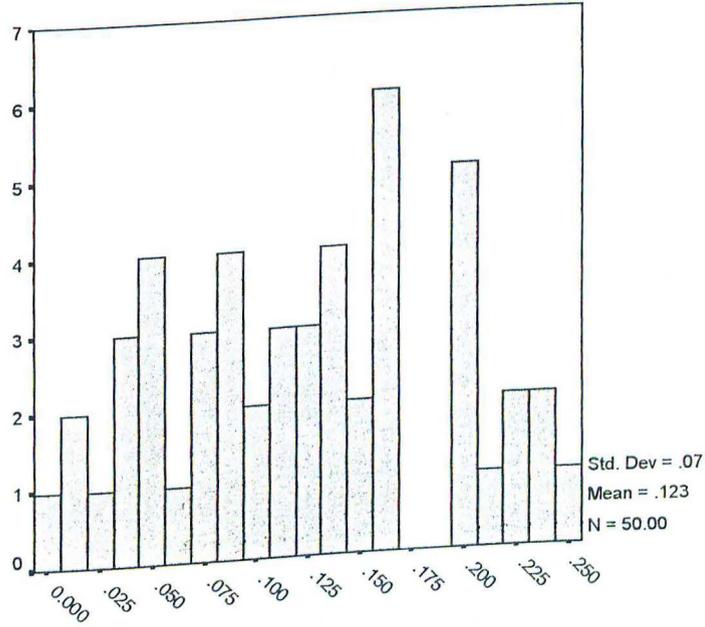


Table B1 - Non-related Item Parameters

Item	a parameter	b parameter	c parameter
1.	1.1657	0.6006	0.0290
2.	1.6241	1.2642	0.0468
3.	1.3494	0.3305	0.0428
4.	1.4049	-1.6621	0.0875
5.	0.9754	1.9284	0.2165
6.	1.3072	1.0653	0.2408
7.	0.9492	0.3461	0.1608
8.	1.2434	-0.8626	0.1144
9.	1.7554	1.9699	0.2397
10.	1.3531	0.5314	0.1283
11.	0.9938	-0.8127	0.1666
12.	0.9709	0.4541	0.0543
13.	1.0605	-2.5556	0.0033
14.	1.0493	1.9800	0.0390
15.	1.2234	2.2956	0.1461
16.	1.4260	1.4830	0.1669
17.	0.8810	-0.9127	0.1164
18.	1.0152	1.3714	0.0134
19.	1.2208	0.2798	0.0422

20.	0.8883	-0.4455	0.1371
21.	1.2458	0.7832	0.1235
22.	1.3512	0.5533	0.0903
23.	0.9461	-1.9199	0.2446
24.	1.0685	-0.3195	0.2016
25.	1.0767	0.6040	0.1255
26.	1.6030	-0.8411	0.1010
27.	1.3111	1.4784	0.2051
28.	2.3059	-1.9287	0.1343
29.	0.8766	0.2594	0.1952
30.	1.7543	-0.2754	0.2203
31.	1.2909	-0.4783	0.0490
32.	1.8583	-0.0270	0.1957
33.	1.0019	-1.3948	0.1099
34.	1.3767	0.6155	0.0596
35.	1.1584	-0.5298	0.1657
36.	2.0313	0.8654	0.0727
37.	1.2320	-0.2496	0.0862
38.	1.1054	-1.7248	0.1676
39.	1.0721	-1.1929	0.1564
40.	0.8469	-1.7308	0.1036
41.	1.5483	-0.0476	0.1459
42.	1.2880	1.1402	0.0753
43.	0.9359	1.0375	0.0512
44.	1.2367	1.4488	0.2238
45.	0.9770	0.1573	0.0769
46.	1.1164	-0.1883	0.0814
47.	1.0944	-1.1361	0.2004
48.	0.9095	-1.5344	0.1316
49.	1.3252	1.2988	0.1398
50.	0.8700	0.2360	0.0071

Figure B4 - Related α parameters

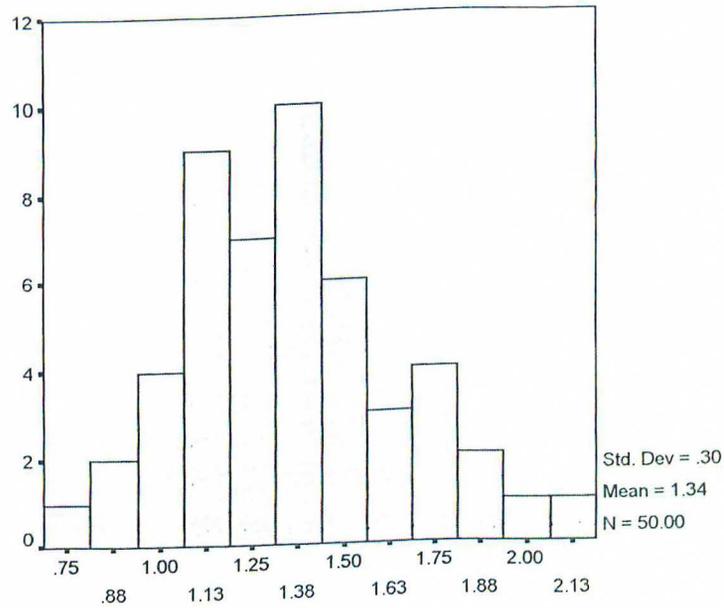


Figure B5 - Related b parameters

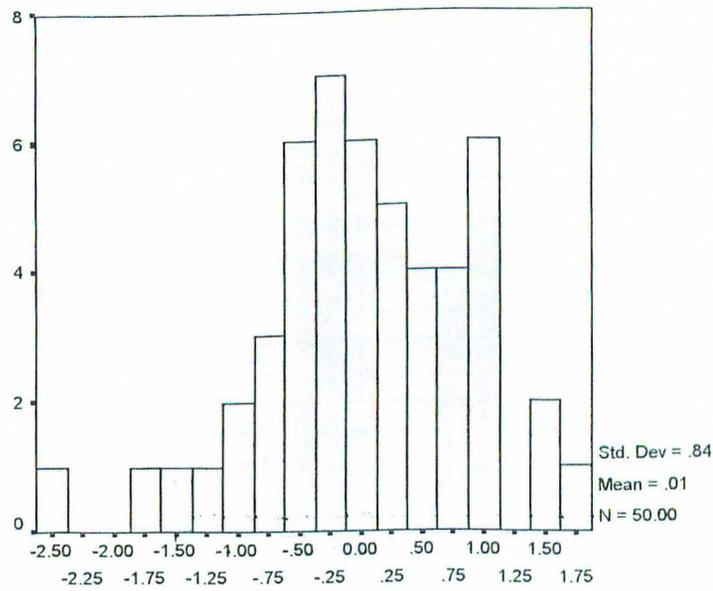


Figure B5 - Related c parameters

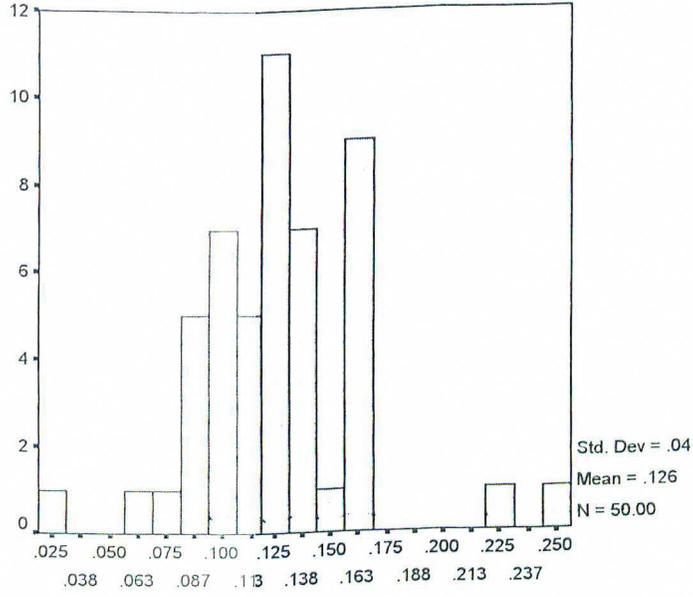


Table B2 - Related a parameters

Item	a parameter	b parameter	c parameter
1.	1.1537	-0.8369	0.1324
2.	1.4342	-0.1768	0.1635
3.	1.0042	0.1307	0.0959
4.	0.8652	1.7188	0.2454
5.	1.2597	-0.3978	0.1262
6.	1.6585	0.9257	0.1091
7.	1.9000	-0.1622	0.0804
8.	1.0035	-1.3171	0.1582
9.	1.3683	0.6424	0.0932
10.	1.1466	-0.1005	0.1206
11.	1.1758	-0.3897	0.1257
12.	1.6123	-0.7450	0.1235
13.	1.4384	-0.0467	0.1291
14.	1.5478	0.0010	0.1299
15.	1.0816	0.2244	0.1131
16.	0.9669	-2.5836	0.0240
17.	1.6989	0.6556	0.1595
18.	1.2150	0.3788	0.0971
19.	1.1211	0.8880	0.1638
20.	1.9474	-1.3883	0.1601
21.	1.3877	-0.1690	0.1071

22.	1.3089	0.6071	0.0834
23.	1.2490	0.6421	0.1675
24.	1.0118	-0.5445	0.0898
25.	1.4735	0.3811	0.1522
26.	1.3597	0.1284	0.1423
27.	1.0756	-0.4138	0.1261
28.	0.7754	1.5237	0.2238
29.	1.2494	-0.1461	0.1040
30.	1.6399	0.9780	0.1228
31.	1.6994	1.0373	0.1611
32.	1.0647	0.0772	0.1428
33.	1.1549	-0.8469	0.1259
34.	1.4443	1.0150	0.1038
35.	1.3609	0.8125	0.0884
36.	1.8272	-0.0927	0.1099
37.	2.0855	0.1777	0.1113
38.	1.0863	0.2397	0.1193
39.	1.6876	-0.9378	0.1611
40.	1.3182	0.0256	0.1389
41.	1.3329	0.5355	0.1030
42.	1.2013	-0.1568	0.1038
43.	0.8304	-1.8110	0.0680
44.	1.3143	0.9498	0.1246
45.	1.8061	-0.2478	0.1429
46.	1.3189	-0.4452	0.0997
47.	1.3408	-0.1297	0.1665
48.	1.5409	1.4589	0.1426
49.	1.2166	-0.5445	0.0869
50.	1.4863	-0.8947	0.1386

Appendix C

Mean values of alpha as a function of # of common items and error level

Figure C1 - Mean values of alpha for Non-related 2PL cases

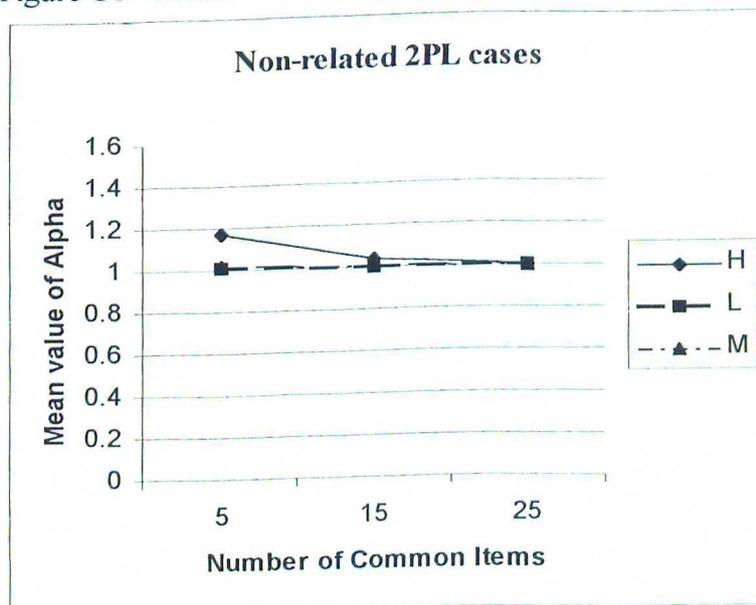


Figure C2 - Mean values of alpha for non-related 3PL cases

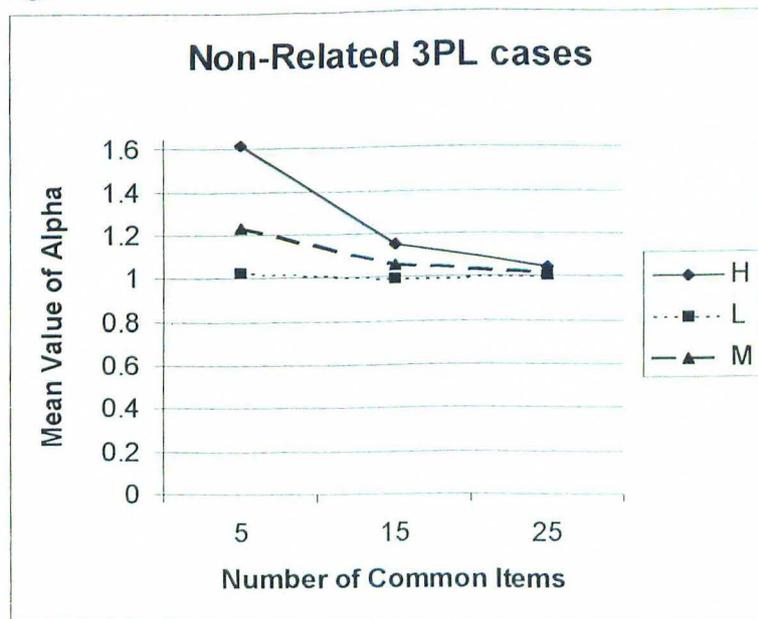


Figure C3 - Mean values of alpha for Related 2PL cases

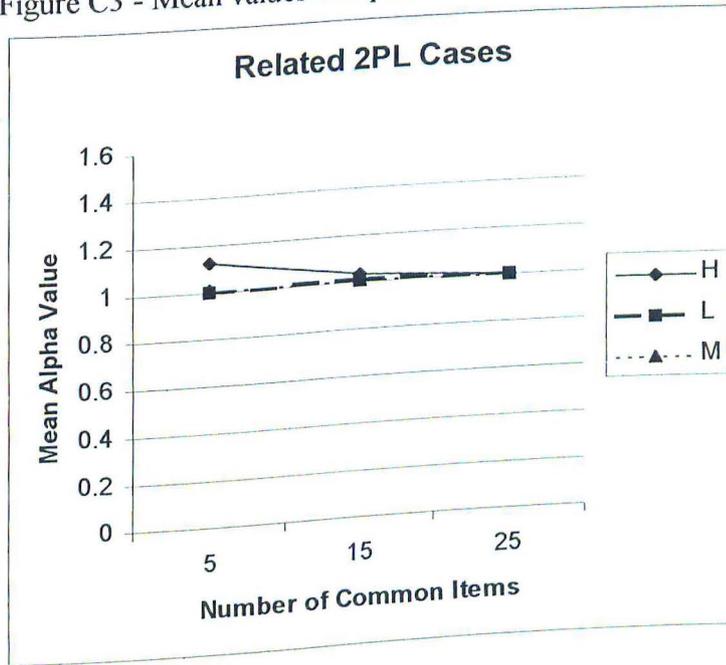
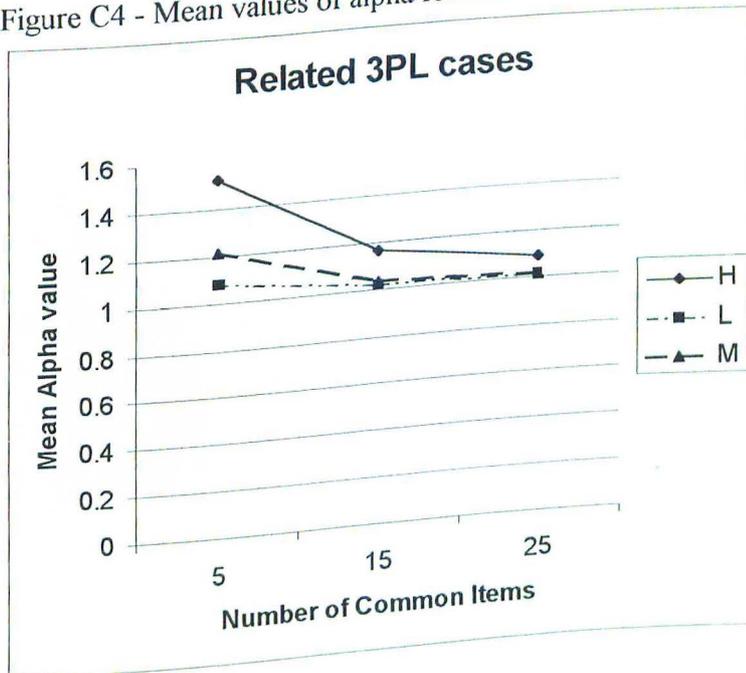


Figure C4 - Mean values of alpha for related 3PL cases



Appendix D

Mean values of beta as a function of # of common items and error level

Figure D1 - Mean values of beta for non-related 2PL cases

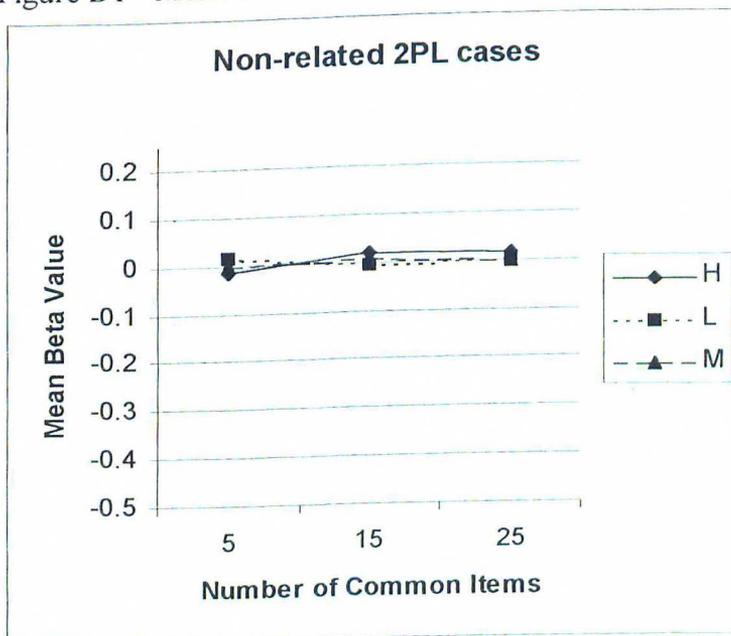


Figure D2 - Mean values of beta for non-related 3PL cases

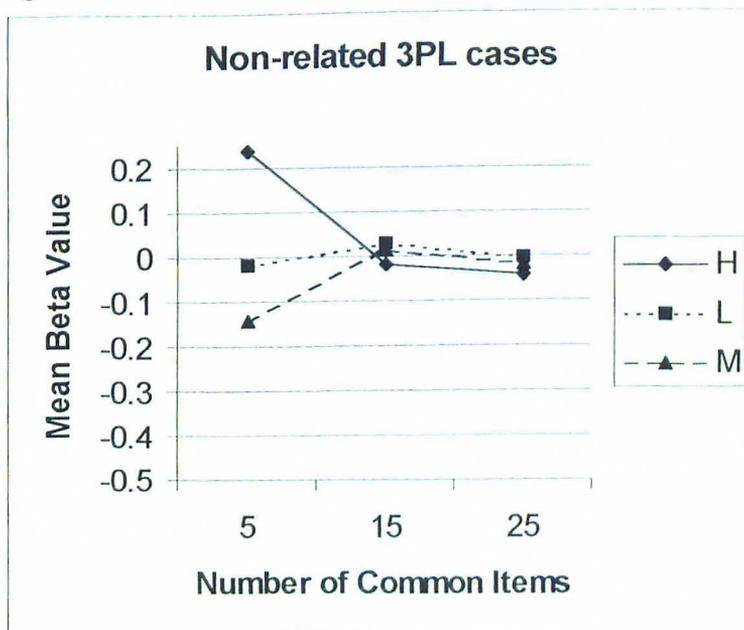


Figure D3 - Mean values of beta for related 2PL cases

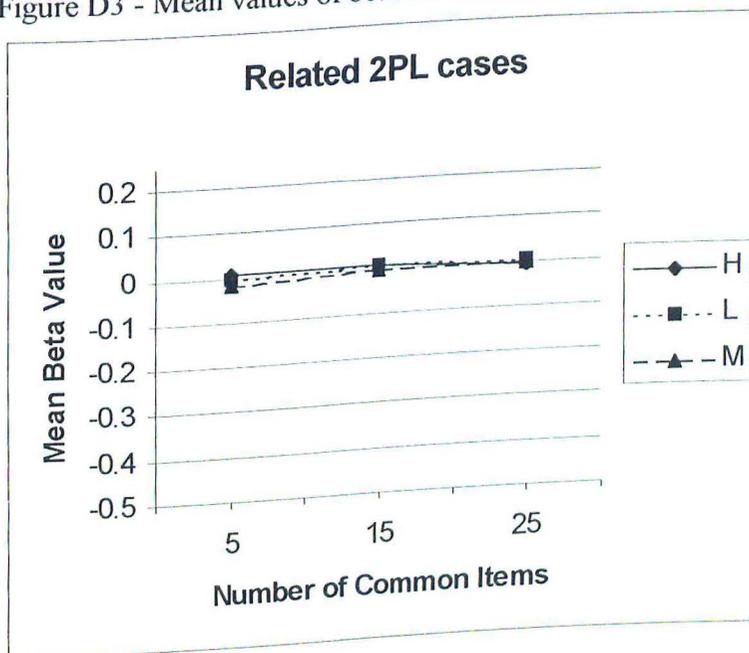
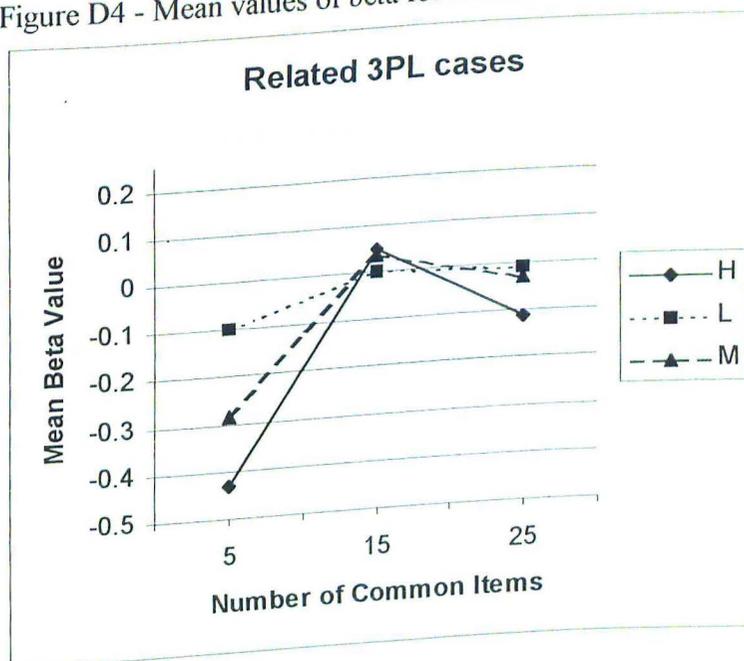


Figure D4 - Mean values of beta for related 3PL cases



Appendix E

RMSE and Bias of alpha as a function of # of common items and error level

Figure E1
RMSE of alpha for non-related 2PL conditions

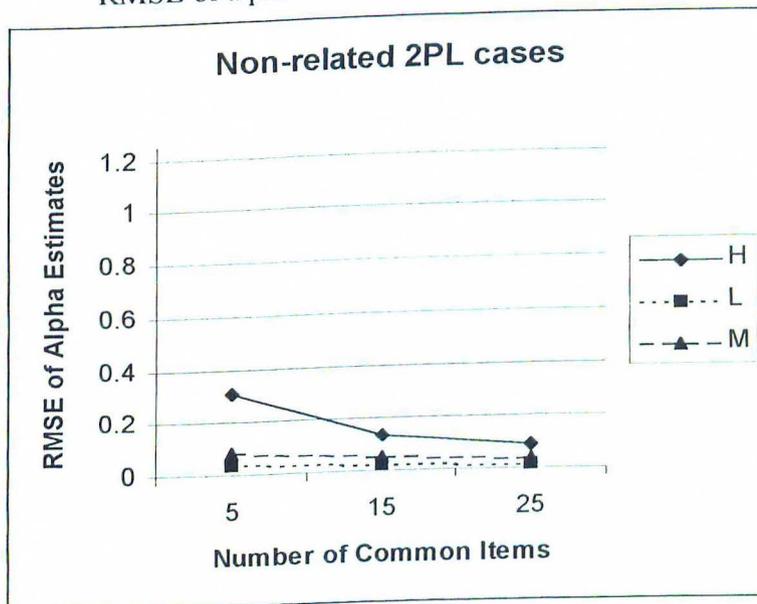


Figure E2
RMSE of alpha for non-related 3PL conditions

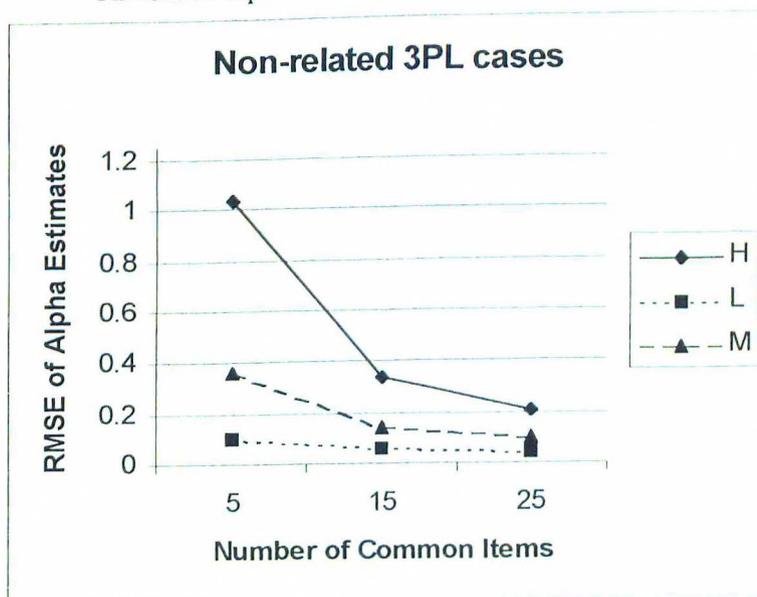


Figure E3
RMSE of alpha for related 2PL conditions

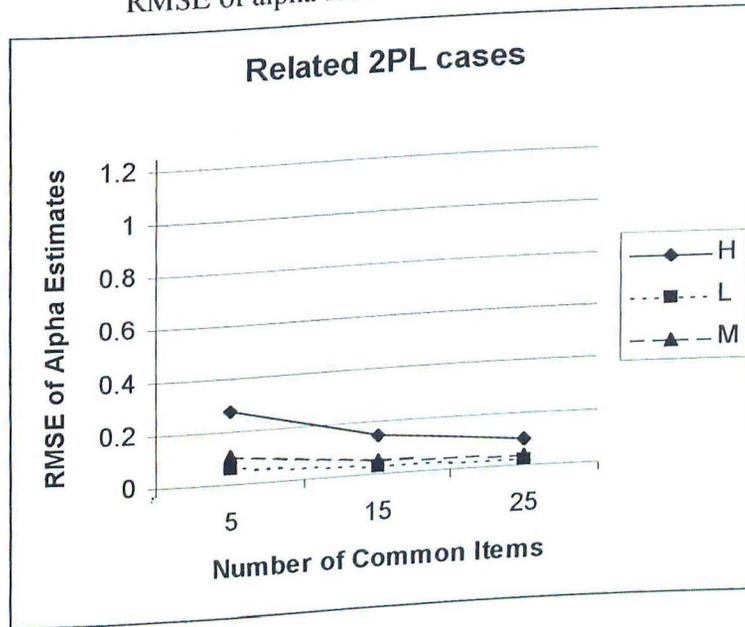


Figure E4
RMSE of alpha for related 3PL conditions

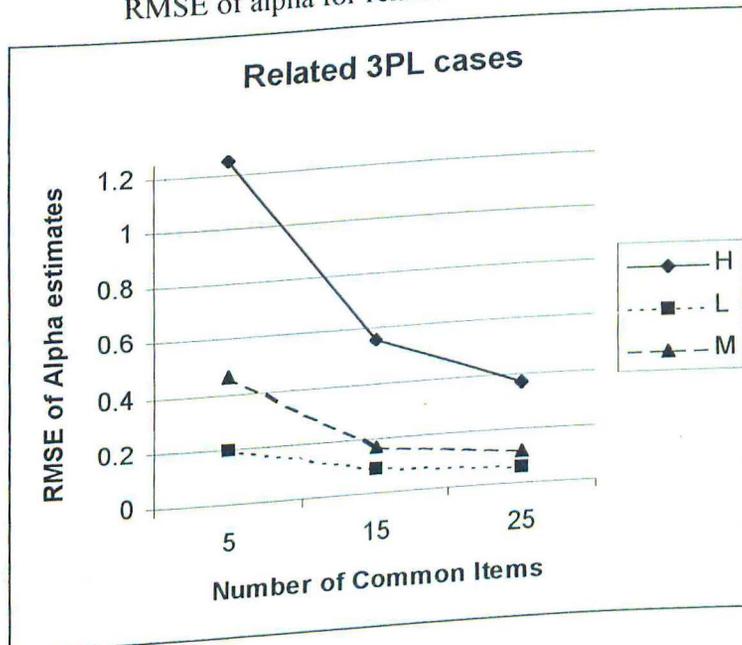


Figure E5
Bias of alpha estimates for non-related 2PL conditions

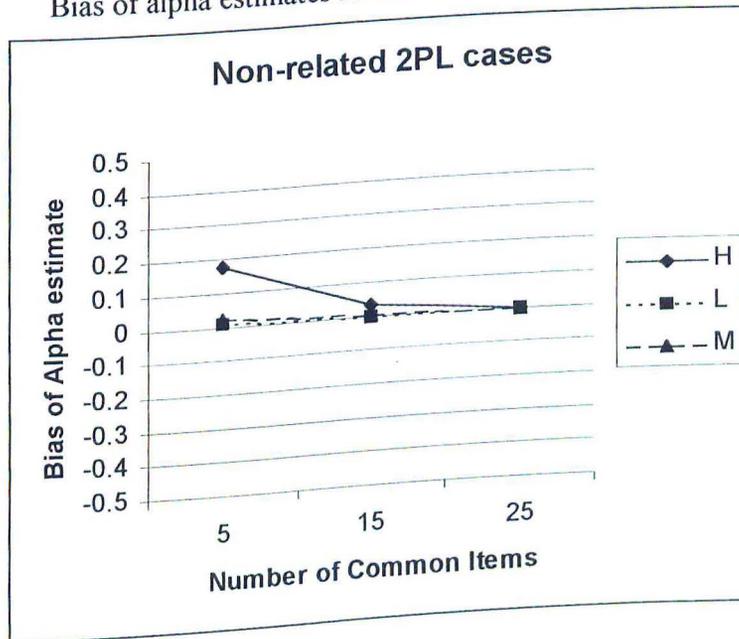


Figure E6
Bias of alpha estimates for non-related 3PL conditions

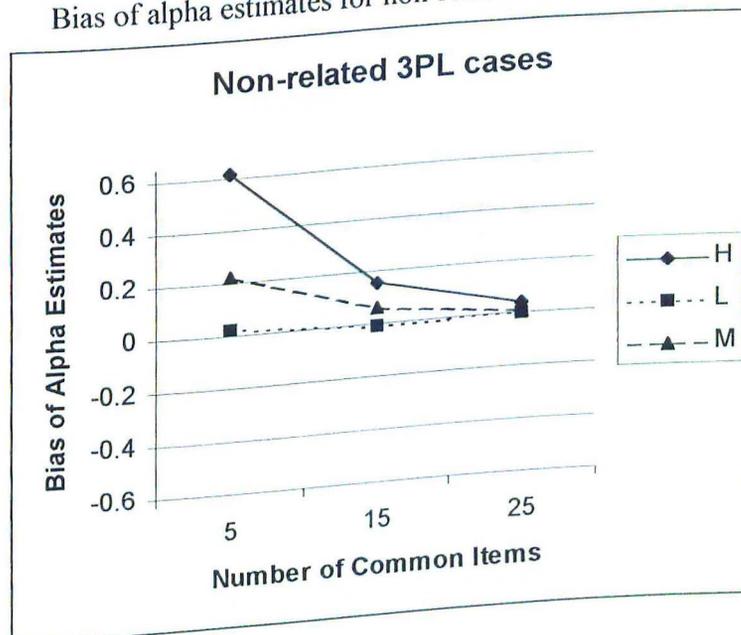


Figure E7
Bias of alpha estimates for related 2PL conditions

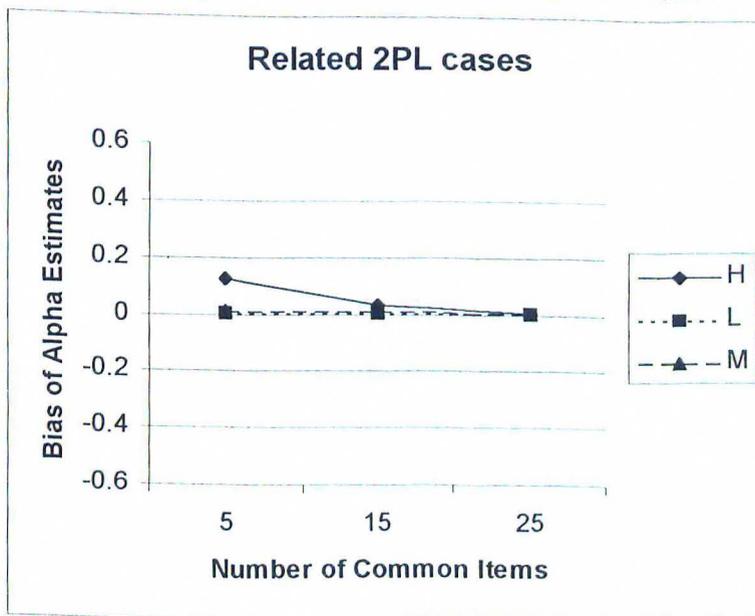
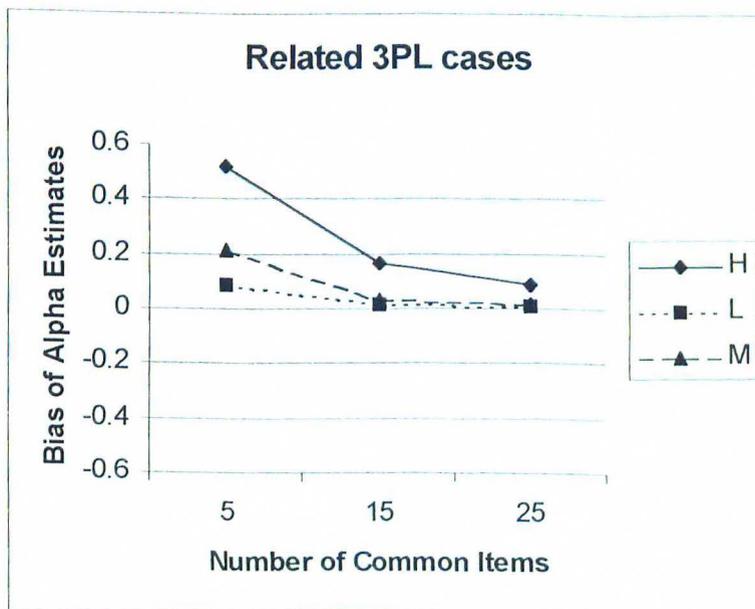


Figure E8
Bias of alpha estimates for related 3PL conditions



Appendix F

RMSE and Bias of beta as a function of # of common items and error level

Figure F1
RMSE of beta estimates for non-related 2PL conditions

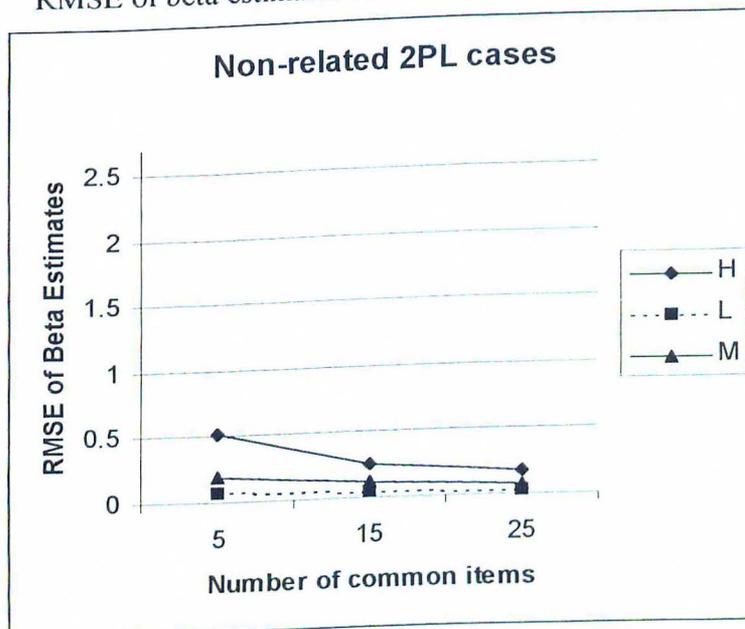


Figure F2
RMSE of beta estimates for non-related 3PL cases

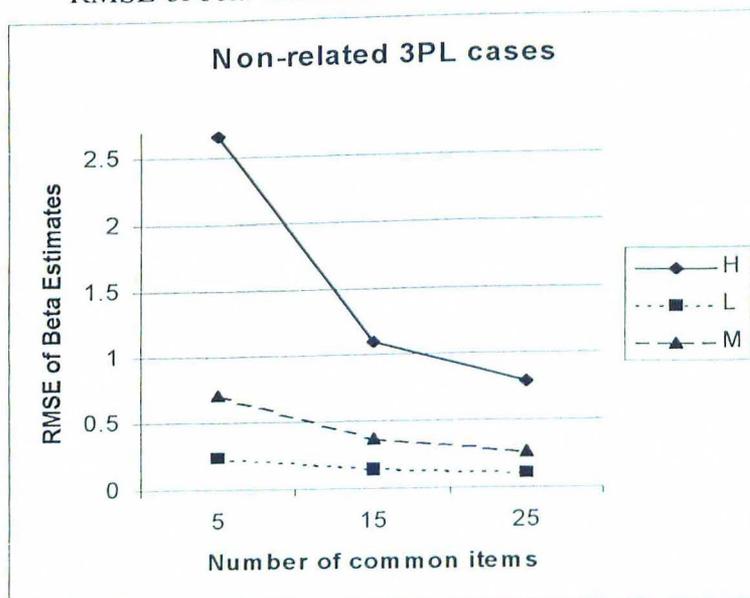


Figure F3
RMSE of beta estimates for related 2PL conditions

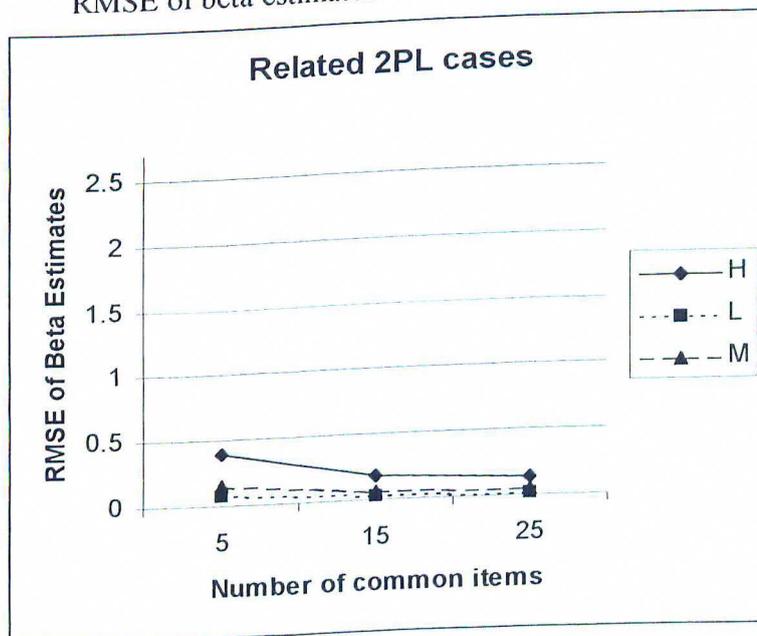


Figure F4
RMSE of beta estimates for related 3PL conditions

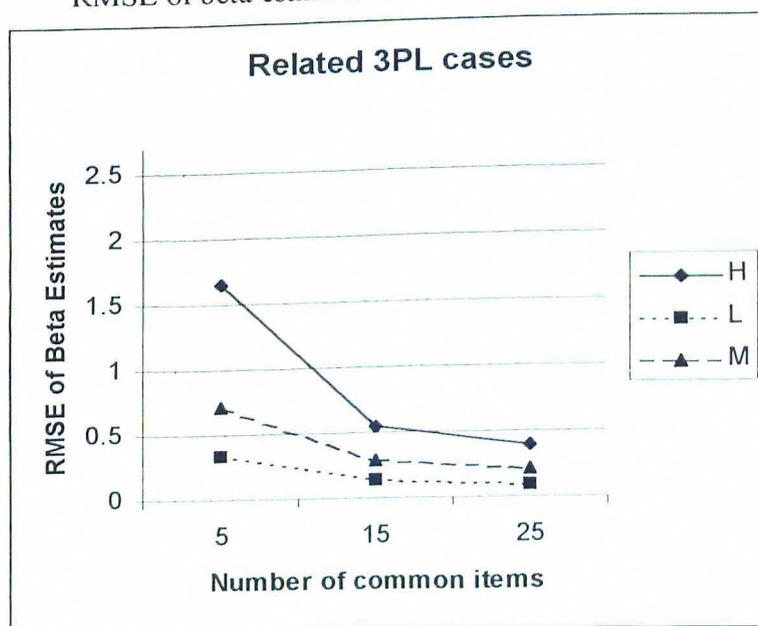


Figure F5
Bias of beta estimates for non-related 2PL conditions

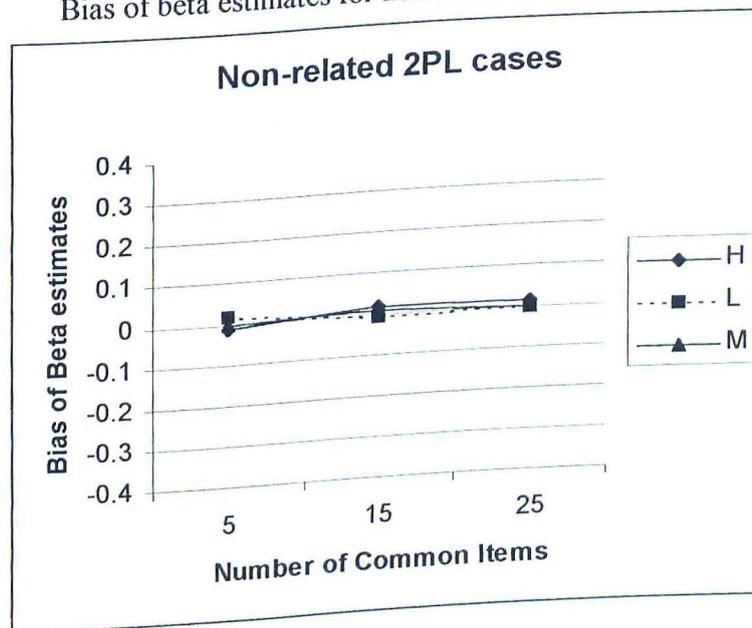


Figure F6
Bias of beta estimates for non-related 3PL conditions

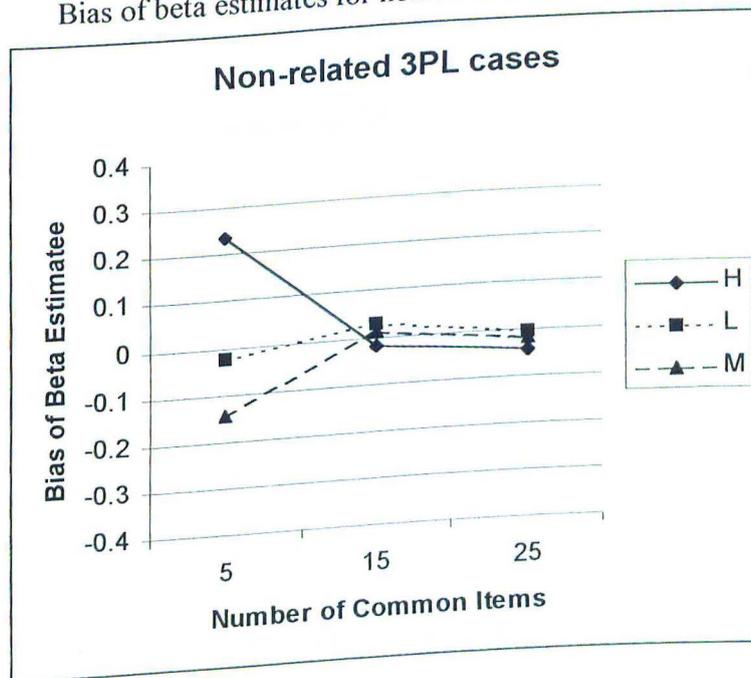


Figure F7
Bias of beta estimates for related 2PL conditions

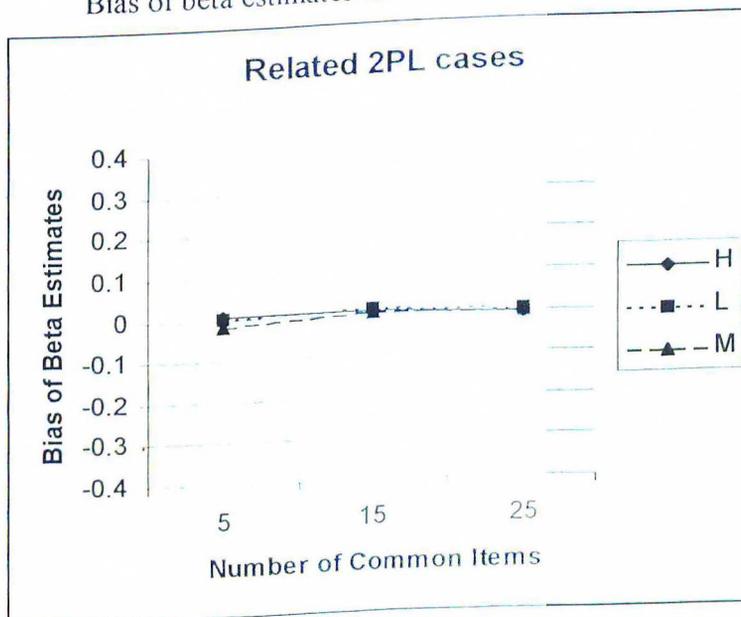
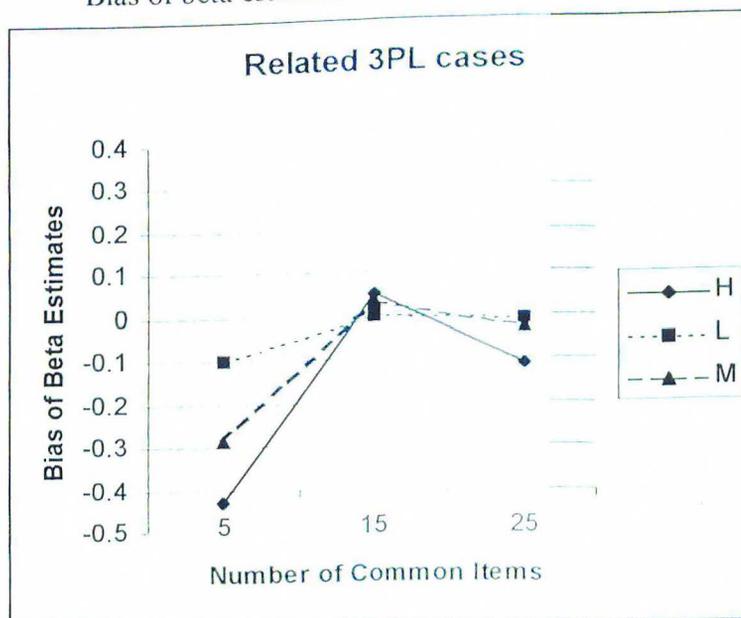


Figure F8
Bias of beta estimates for related 3PL conditions



Appendix G

Mean Improvement Ratios (IR) as a function of # of common items and error level

Figure G1
Mean Improvement Ratio for non-related 2PL conditions

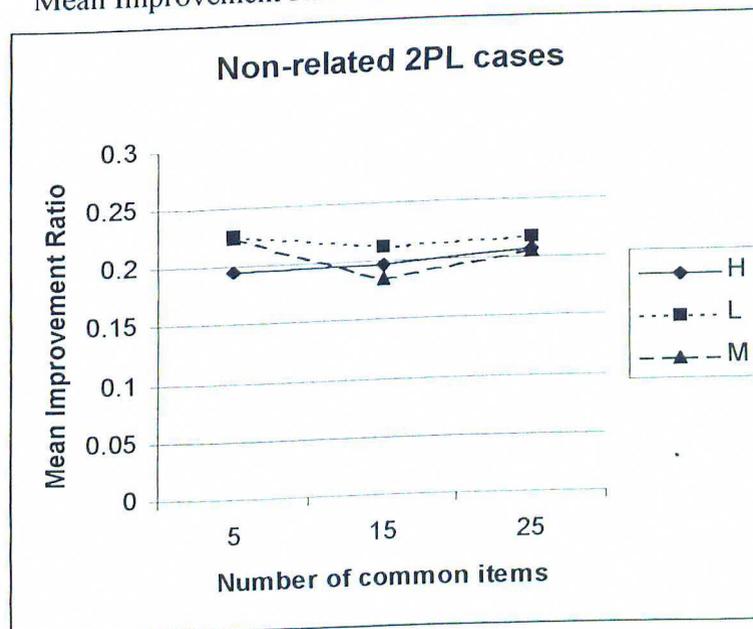


Figure G2
Mean Improvement Ratios for non-related 3PL conditions

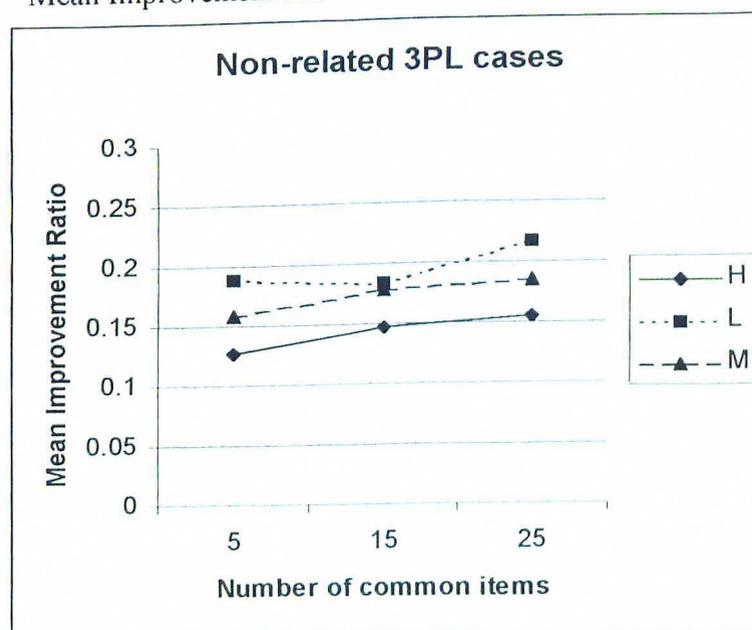


Figure G3
Mean Improvement Ratios for related 2PL conditions

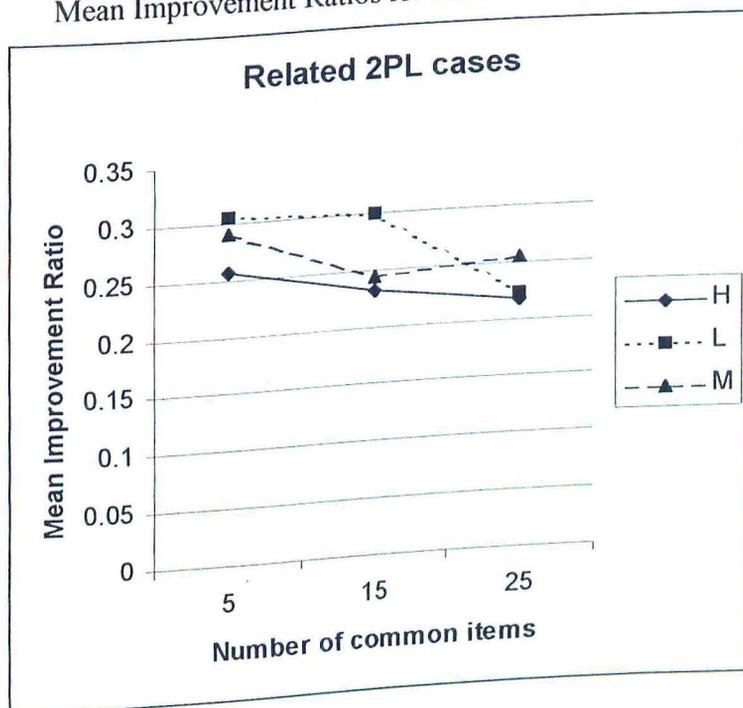
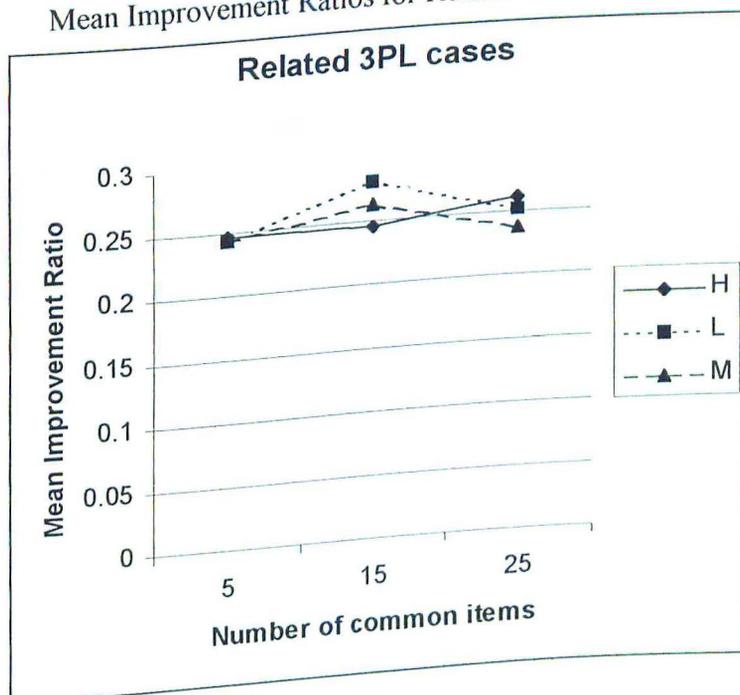


Figure G4
Mean Improvement Ratios for Related 3PL conditions



Appendix H

Mean F_{\max} values as a function of # of common items and error level

Figure H1 - Mean F_{\max} for non-related 2PL conditions

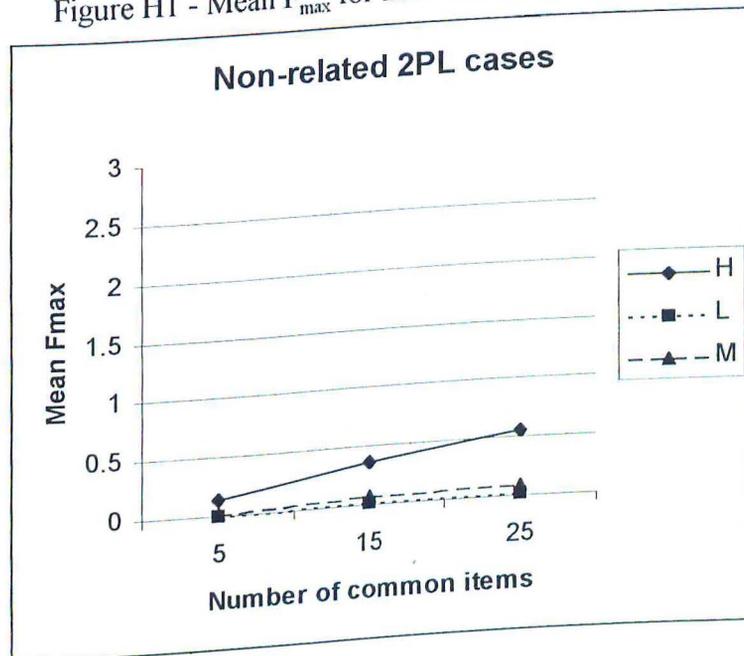


Figure H2 - Mean F_{\max} for non-related 3PL conditions

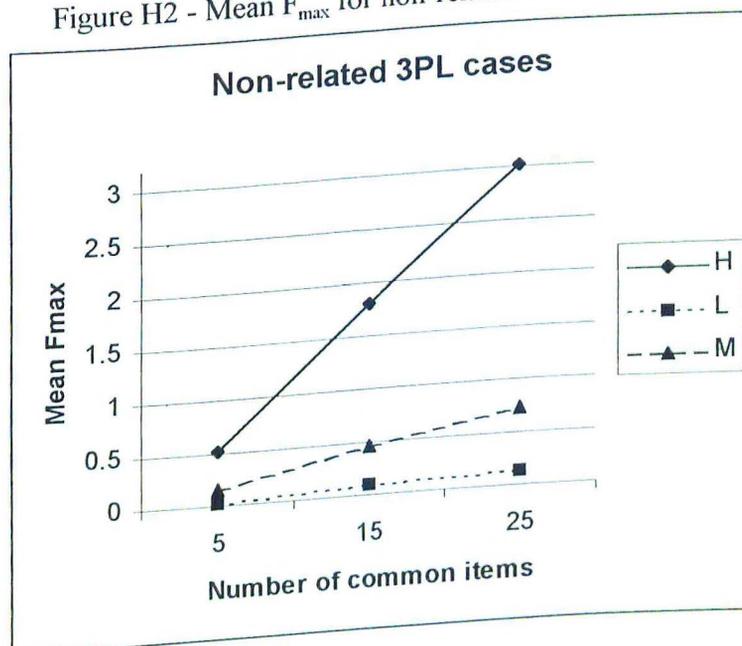
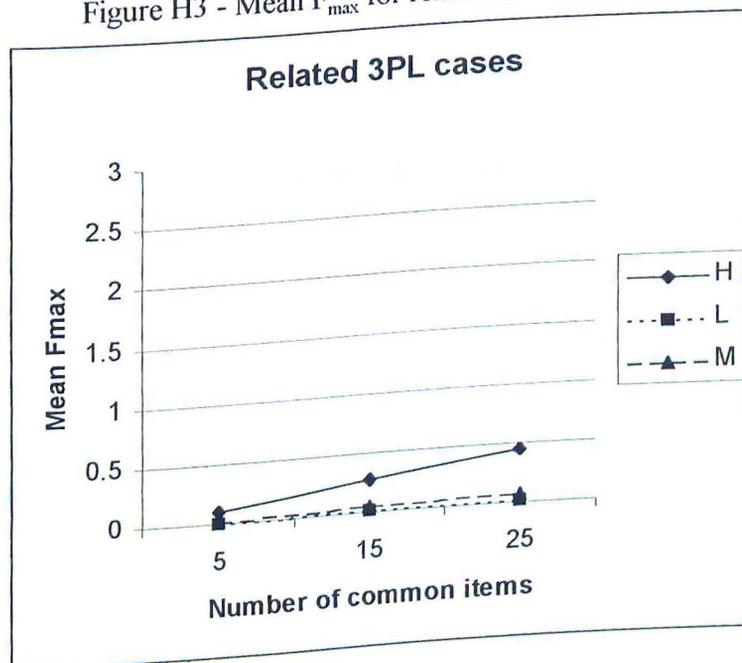
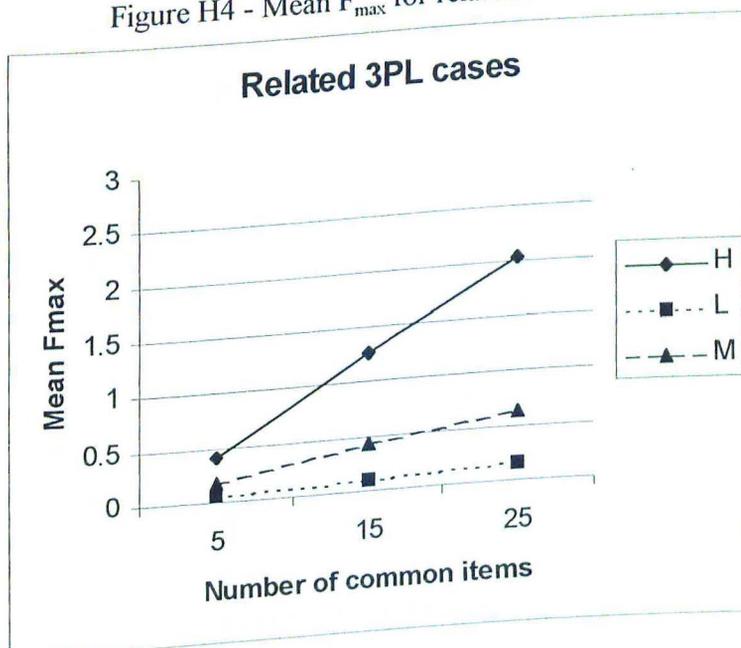


Figure H3 - Mean F_{\max} for related 2PL conditionsFigure H4 - Mean F_{\max} for related 3PL cases

Appendix I

Mean F_{converge} values as a function of # of common items and error level

Figure 11 - Mean F_{converge} for non-related 2PL conditions

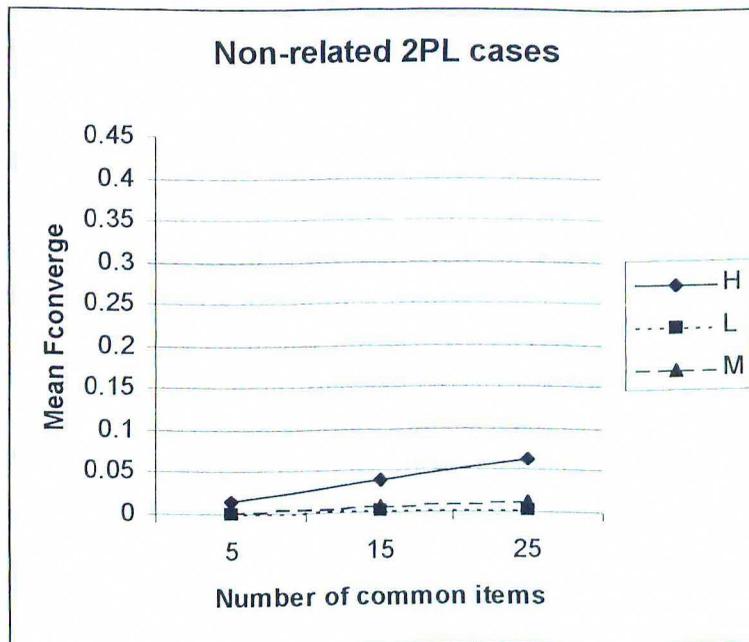


Figure 12 - Mean F_{converge} for non-related 3PL conditions

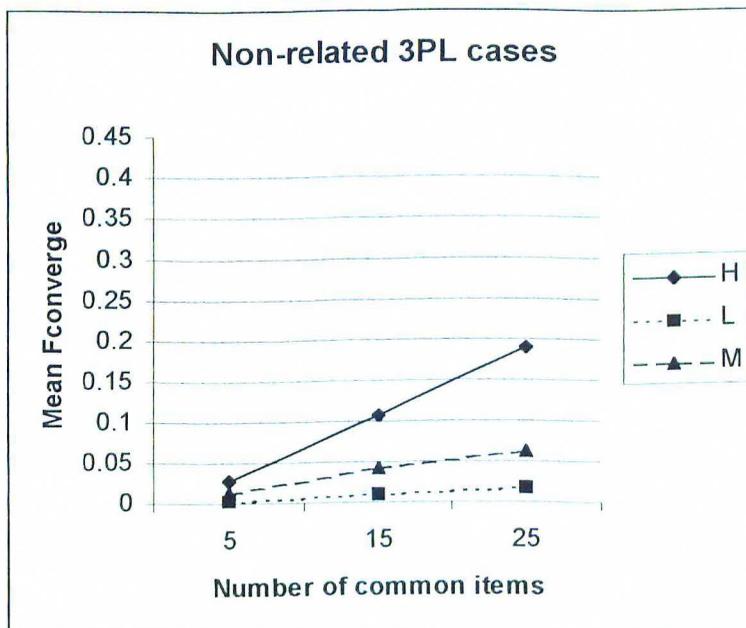
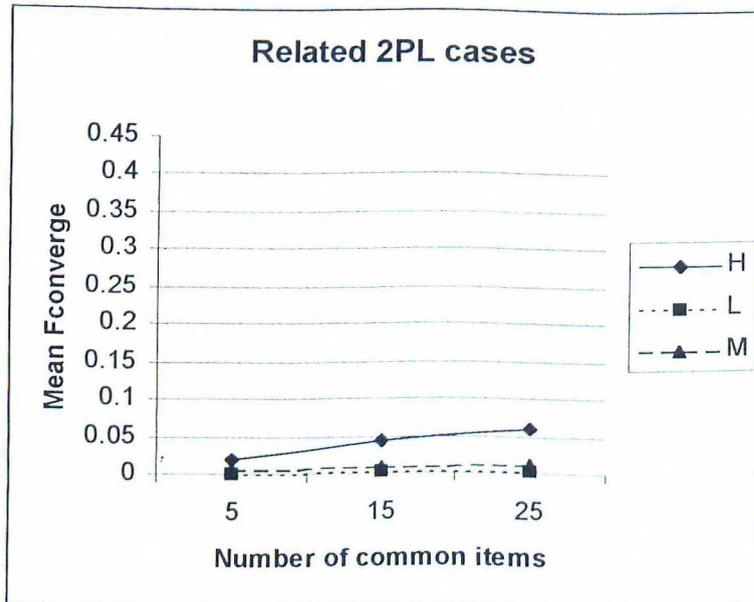
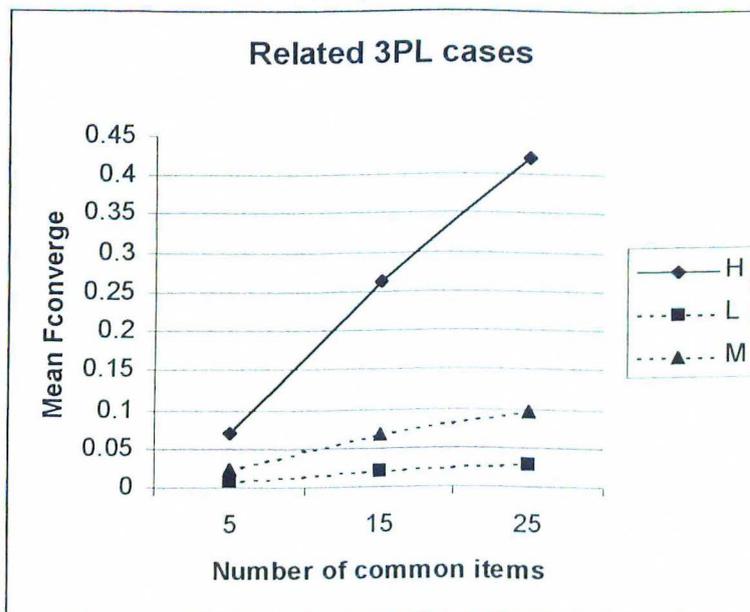


Figure I3 - Mean $F_{converge}$ for related 2PL conditionsFigure I4 - Mean $F_{converge}$ for related 3PL conditions

Appendix J

Mean Theta RMSE values as a function of # of common items and error level

Figure J1
Mean Theta RMSE values for non-related 2PL conditions

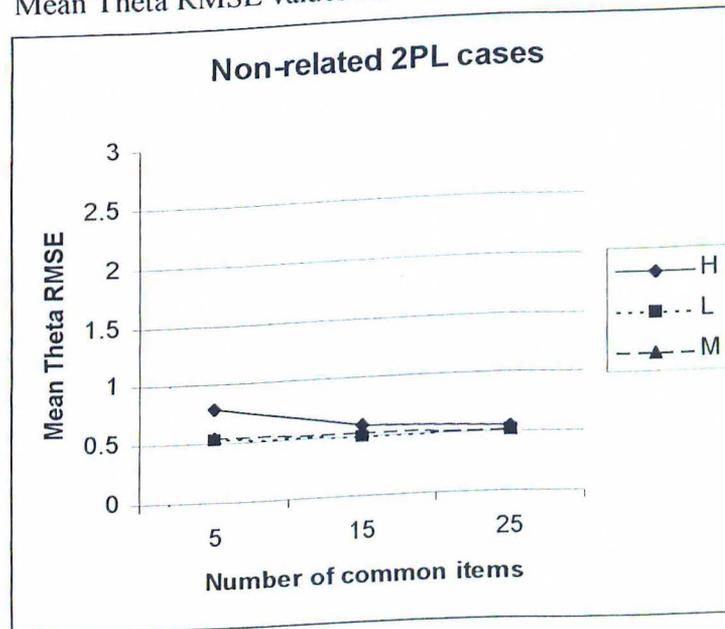


Figure J2
Mean Theta RMSE values for non-related 3PL conditions

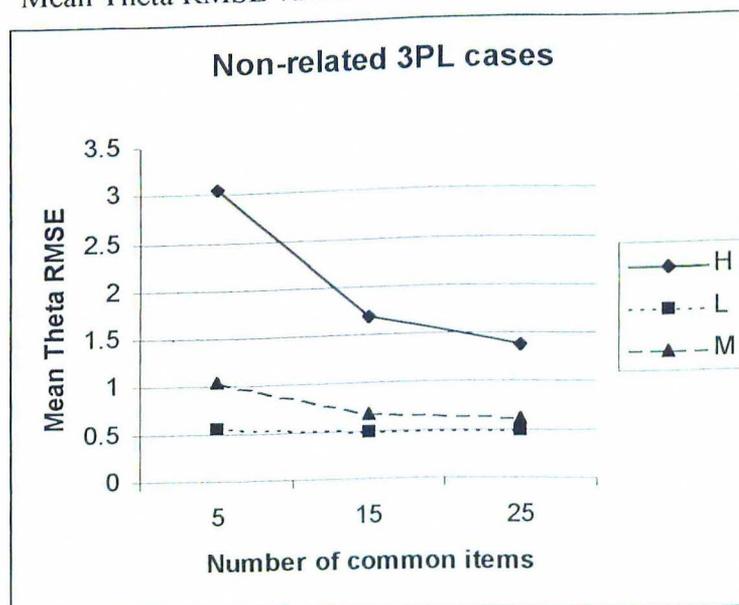


Figure J3
Mean Theta RMSE values for related 2PL conditions

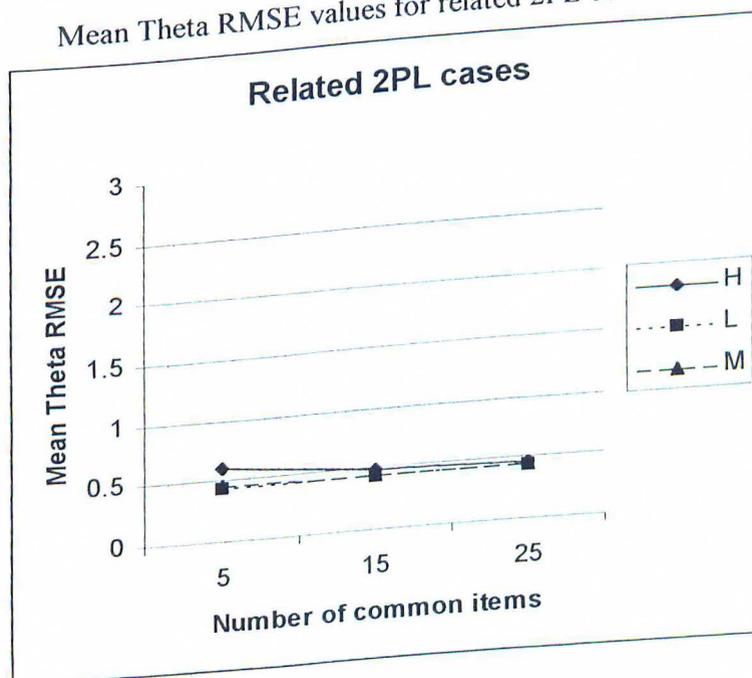
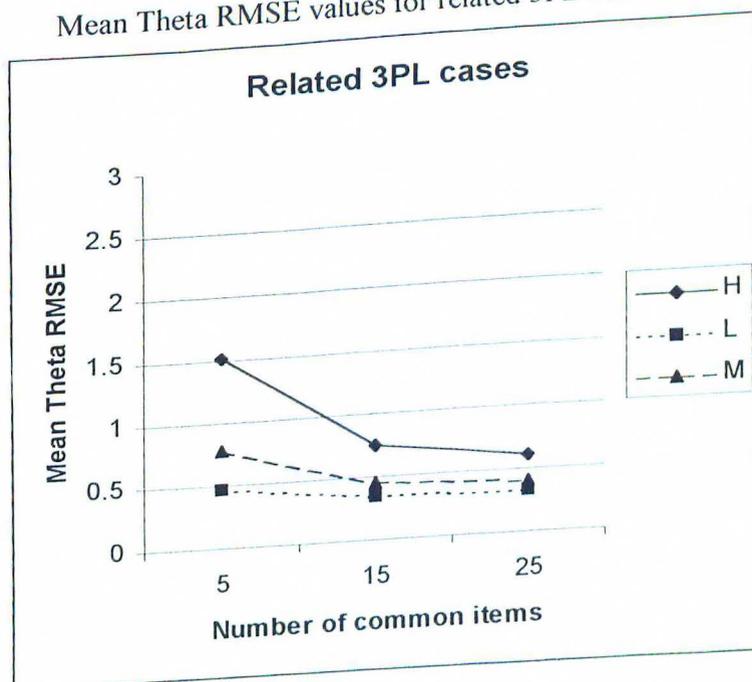


Figure J4
Mean Theta RMSE values for related 3PL conditions



Appendix K

Mean True Score RMSE values as a function of # of common items and error level

Figure K1
Mean True Score RMSEs for non-related 2PL conditions

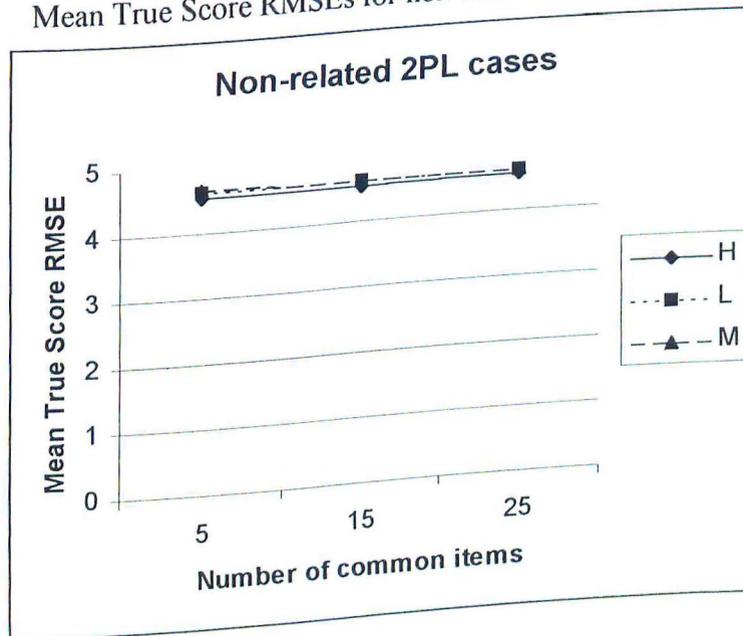


Figure K2
Mean True Score RMSEs for non-related 3PL conditions

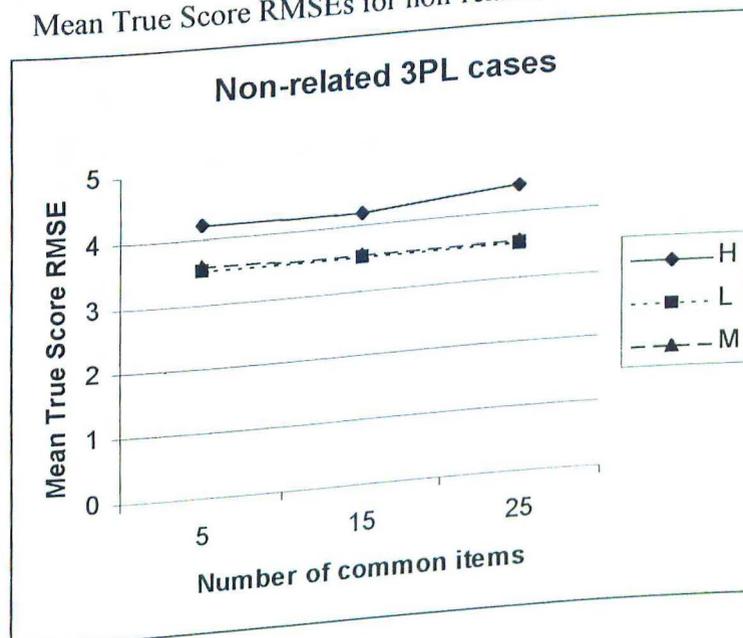


Figure K3
Mean True Score RMSEs for related 2PL conditions

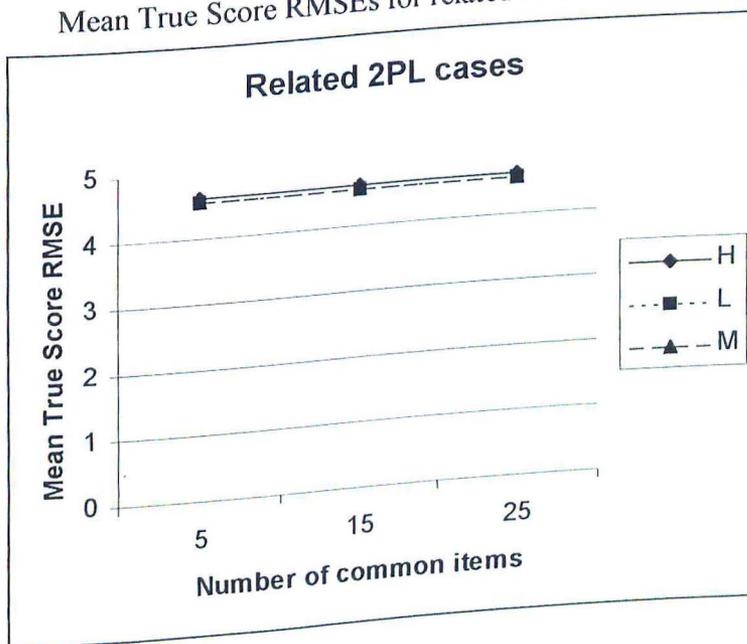
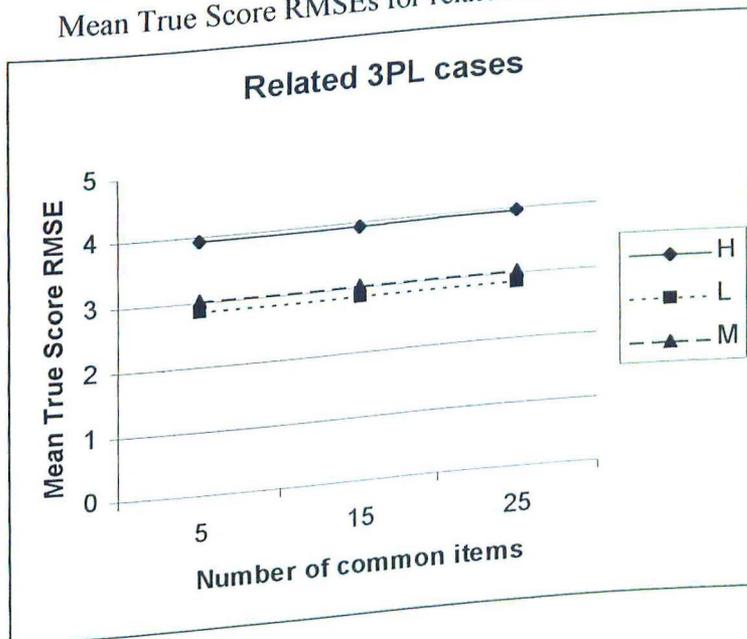


Figure K4
Mean True Score RMSEs for related 3PL conditions



Appendix L

Source code listing

```

#define pi 3.141592654
/* values needed throughout program; chosen in proposal */
/* for EAP estimation */
#define numQuads 15
#define numExaminees 1000 /* # of examinees */
/* # of total items */
#define numItems 50
/* max # of common items (i.e., 5, 15, 25) */
#define maxCommon 25
/* for F calculations */
#define GPOINTS 99
/* Used for F Calculations */
#define MAXITEMS 50
/* Should correspond to highest # of thetas */
#define MAXTHETAS 1000

/* Mean and variance of a parameter distribution */
#define MEANA 1.25
#define VARA .09

/* values to define regions of relation for a and b params */
#define LOWA 1.0
#define LOWB -1.5
#define LOWC .08
#define HIGHA 1.75
#define HIGHB 1.5
#define HIGHC .17

#define minA .8
#define maxA 2.5

/* non-related error conditions */
#define LOWA2 .07
#define MODA2 .15
#define HIA2 .38
#define LOWB2 .11
#define MODB2 .25
#define HIB2 .63

#define LOWA3 .10
#define MODA3 .23
#define HIA3 .53
#define LOWB3 .30
#define MODB3 .91

```

```

#define HIB3          2.73

#define LOWCERR .05
#define MEDCERR .10
#define HICERR .20

/* definitions for random number generation */
#define IA 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS).

/*
** Define function prototypes
*/
void createParms(int, char *, long *, int);
float ran1(long *);
float gasdev(long *);
float logNormal(long *);
char *makeFName(char, char, char, char *, int, char *);
void genAnswers(int, int, char *, char *, char *, long *);
void calcQuadPts(float, float, int, char *);
void getWeights(float, float, char *);
void eap(char *,
         char *,
         char *,
         char *,
         int, int,
         char *);
double stdErr(char *, char *);
void genRelate(int, char *, char *, char *, char *, float[numItems][3], long *, int);
void genItems(int, int, char *, char *, float [numItems][3], long *);
void hpsort(unsigned long, float *);
void piksrt(int, float *);
void piksrt2(int, float [numItems][3], float [numItems][3]);
void pickCommon(float [numItems][3], int, float [maxCommon][3]);

```

```
double probRight(float, float, float, float, int, int);
double square(double);
void setGPoints(char *);
double fMax(char *, char *, char *, int, int, int);
double expectedScore(float [MAXITEMS], float [MAXITEMS],
                    float [MAXITEMS], int, float, int, int);

double sumScores(char *, char *, int, int, int, int);
void trueScores(char *, char *, char *, int, int, int, int);
double FVal(char *, char *, char *, int, int, int, int);
void stripEquate(char *, char *, char *, char *, char *); /* strip values out of
EQUATE output */
void makeEquate(char *, char *, char *, char *, char *, char *, int, int);
double Dvalue(char *, char *);
```

```

/*
**      File Name:      irtstudy.cpp
**      Created:       01/98
**      Author:        Gary Kaskowitz
**                   Department of Measurement, Statistics
**                   and Evaluation
**                   University of Maryland
**
**      Description:   Main driver routine for study. This program
**                   controls flow of simulation and is bundled
**                   with the modified random number generators.
**
*/

```

```
#include "defs.h"
```

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <string.h>
```

```

/*
** Define common variables
*/

```

```

int newCall = 1;      /* identify whether main routine being called fresh */
static long iy=0;
static long iv[NTAB];
static char ranVals[80];
long lastSeed=-1;
int trueDone = 0;

```

```

float initItems[numItems][3];      /* initial form item estimates */
float tgtItems[numItems][3];      /* target form item estimates */
float initCommon[maxCommon][3];   /* initial form common items */
float tgtCommon[maxCommon][3];   /* target form common items */

```

```

main(int argc, char *argv[])
{

```

```

    FILE *myout;
    FILE *ranPtr;
    FILE *fFile;
    FILE *seTrueFile;

```

```
FILE *seThetaFile;
FILE *newPtr;
FILE *initPtr;

char SE, Model, Relate;          /* skip parameter initialization? */
char skipParm;
double seError=0;
char NumCom[3], Path[20];
char cmnPath[20];
int i;
int twoPL = 0, threePL = 0;
int makeRelate = 0;
int level;
int loop;

char nameit[80];
char itemEstFile[80];
char wgtFile[80];
char quadFile[80];
char answerFile[80];
char passFile[80];
char parmFile[80];
char abilsFile[80];
char estimsFile[80];
char frEstFile[80];
char toEstFile[80];
char FpointsFile[80];
char initFile[80];
char tgtFile[80];
char newFile[80];
char Relatea[80];
char Relateb[80];
char initCmnFile[80];
char tgtCmnFile[80];
char frTrueFile[80];
char toTrueFile[80];
char nuTrueFile[80];
char prmTrueFile[80];
char conTrueFile[80];
char runFile[80];
char newThetas[80];
char inEquate[80];
char outEquate[80];
```

```

char newCommon[80];
char cmdString[80];
char fVals[80];
char seVals[80];
char sortInit[80];
char sortTgt[80];

char acoeffStr[20], kcoeffStr[20], fcoeffStr[20];
float acoeff=0, kcoeff=0, fcoeff=0;
float avalInit, bvalInit, cvalInit;
float avalNew, bvalNew;

float aNew[maxCommon], bNew[maxCommon]; /* used for EQUATE file
conversion */
double Fmax = 0;
double Fconv = 0;
double ImpRatio =0;

int numCommon=0;

FILE *initSort;
FILE *tgtSort;
FILE *initCmn;
FILE *tgtCmn;
FILE *cmnPtr;

memset(nameit,0,sizeof(nameit));
memset(NumCom, 0, sizeof(NumCom));
memset(Path, 0, sizeof(Path));
memset(passFile,0,sizeof(passFile));
memset(cmnPath, 0, sizeof(cmnPath));

/*
** Set the common path for storage of programs and constant
** parameters across all cells. Hard coded for now, probably
** not the best way to do this.
*/
strcpy(cmnPath, "d:\\data\\");

/*
** First, let's read in the routine parameters which define
** the cell of interest.
*/

```

```

SE = *argv[1];
Model = *argv[2];
Relate = *argv[3];
strcpy(NumCom, argv[4]);
strcpy(Path, argv[5]);
skipParm = *argv[6];

/*
** Name random values file
*/
strcpy(ranVals, cmnPath);          /* want to use across cells */
strcat(ranVals, "random.num");

/*
** Update necessary variables for setting proper options
*/
if (SE == 'L' || SE == 'l')
    level = 1;
else if (SE == 'M' || SE == 'm')
    level = 2;
else if (SE == 'H' || SE == 'h')
    level = 3;

if (Model == '2')
{
    twoPL = 1;
    threePL = 0;
}
else if (Model == '3')
{
    twoPL = 0;
    threePL = 1;
}

if (Relate == 'y' || Relate == 'Y')
    makeRelate = 1;
else
    makeRelate = 0;

memset(fVals, 0, sizeof(fVals));
strcpy(fVals, Path);
strcat(fVals, "Fvalues.out");

```

```

    memset(seVals, 0, sizeof(seVals));
    strcpy(seVals, Path);
    strcat(seVals, "trueErr.out");

    fFile = fopen(fVals, "w");
    seTrueFile = fopen(seVals, "w");

    memset(seVals, 0, sizeof(seVals));
    strcpy(seVals, Path);
    strcat(seVals, "thetaErr.out");

    fFile = fopen(fVals, "w");
    seThetaFile = fopen(seVals, "w");

    /*
    ** Initialize the random number generators
    */
    if (skipParm == 'n' || skipParm == 'N')
    {
        ranl(&lastSeed);
        gasdev(&lastSeed);
        logNormal(&lastSeed);
    }
    /*
    ** Make the appropriate filename based upon what we cell
    ** we are in.
    */

    memset(parmFile, 0, sizeof(parmFile));
    strcpy(parmFile, cmnPath);
    if (makeRelate)
        strcat(parmFile, "related.prm");
    else
        strcat(parmFile, "nonrel.prm");
    if (skipParm == 'n' || skipParm == 'N')
        createParams(makeRelate, parFile, &lastSeed, twoPL);

    memset(quadFile, 0, sizeof(quadFile));
    strcpy(quadFile, cmnPath);
    strcat(quadFile, "quads.qd");
    if (skipParm == 'n' || skipParm == 'N')
        calcQuadPts(-4.0, 4.0, numQuads, quadFile);

```

```

strcpy(abilsFile, cmnPath);
strcat(abilsFile, "abils.abi");
strcpy(answerFile, cmnPath);
strcat(answerFile, "answers.dat");
if (skipParm == 'n' || skipParm == 'N')
    genAnswers(twoPL,threePL,parmFile,abilsFile,answerFile,&lastSeed);

```

```

strcpy(wgtFile, cmnPath);
strcat(wgtFile, "weights.wgt");
if (skipParm == 'n' || skipParm == 'N')
    getWeights(-4.0, 4.0, wgtFile);

```

```

/*
** Main loop of program
*/
for (i=0; i<=249; i++)
{
    printf("\n\nLoop %d\n", i);
    memset(initFile, 0, sizeof(initFile));
    strcpy(initFile, Path);
    strcat(initFile, makeFName(SE,Model,Relate,NumCom,i, "ini"));

```

```

    memset(tgtFile, 0, sizeof(tgtFile));
    strcpy(tgtFile, Path);
    strcat(tgtFile, makeFName(SE,Model,Relate,NumCom,i, "tgt"));

```

```

if (!makeRelate)

```

```

{
    /* Generate items for the initial form */
    genItems(level,twoPL, parmFile,
             initFile, initItems,
             &lastSeed);

```

```

    /* Now, generate items for the target form */
    genItems(level,twoPL, parmFile,
             tgtFile, tgtItems,
             &lastSeed);

```

```

}
else if (makeRelate)

```

```

{
    strcpy(Relatea, cmnPath);
    if (twoPL)

```

```

/* same across cells */

```

```

        strcat(Relatea, "relateda.2pl");
    else
        strcat(Relatea, "relateda.3pl");

    strcpy(Relateb, cmnPath);          /* same across cells */
    if (twoPL)
        strcat(Relateb, "relatedb.2pl");
    else
        strcat(Relateb, "relatedb.3pl");

    /* first, generate the items for the initial form */
    genRelate(level, Relatea, Relateb,
              parmFile,
              initFile,
              initItems,
              &lastSeed,
              twoPL);

    /* now generate the items for the target form */
    genRelate(level, Relatea, Relateb,
              parmFile,
              tgtFile,
              tgtItems,
              &lastSeed,
              twoPL);

}

/*
** Now, we will estimate the thetas for the initial form given
** the answers generated from the parameters.
*/

memset(frEstFile, 0, sizeof(frEstFile));
strcpy(frEstFile, Path);
strcat(frEstFile, makeFName(SE, Model, Relate, NumCom, i, "qfm"));

eap(initFile,
    wgtFile,
    quadFile,
    answerFile,
    twoPL, threePL,
    frEstFile);

```

```

/*
** And then we will estimate the thetas for the target
** files, using the same set of generated answers.
*/

memset(toEstFile, 0, sizeof(toEstFile));
strcpy(toEstFile, Path);
strcat(toEstFile, makeFName(SE,Model,Relate,NumCom,i,"qto"));

eap(tgtFile,
    wgtFile,
    quadFile,
    answerFile,
    twoPL, threePL,
    toEstFile);

/*
** After the items have been generated, we can pick the common
** ones which will be used for the equating. However, before
** we do that, the items in the two forms must first be sorted.
*/
/*
** First, sort the items simultaneously, using the target items
** to determine the sort order.
*/
piksr2(numItems, tgtItems, initItems);

memset(sortInit, 0, sizeof(sortInit));
strcpy(sortInit, Path);
strcat(sortInit, "sorted.ini");

memset(sortTgt, 0, sizeof(sortTgt));
strcpy(sortTgt, Path);
strcat(sortTgt, "sorted.tgt");

initSort = fopen(sortInit, "w");
tgtSort = fopen(sortTgt, "w");
for (loop=0; loop<numItems; loop++)
{
    fprintf(initSort, "%f\t%f\t%f\n", initItems[loop][0],
initItems[loop][1],initItems[loop][2]);
    fprintf(tgtSort, "%f\t%f\t%f\n", tgtItems[loop][0],
tgtItems[loop][1],tgtItems[loop][2]);
}

```

```

    }
    fclose(initSort);
    fclose(tgtSort);

    /* select the common items from each form to be used for equating */
    numCommon = atoi(NumCom);
    pickCommon(initItems, numCommon, initCommon);
    pickCommon(tgtItems, numCommon, tgtCommon);

    memset(initCmnFile, 0, sizeof(initCmnFile));
    strcpy(initCmnFile, Path);
    strcat(initCmnFile, "common.ini");

    memset(tgtCmnFile, 0, sizeof(tgtCmnFile));
    strcpy(tgtCmnFile, Path);
    strcat(tgtCmnFile, "common.tgt");

    initCmn = fopen(initCmnFile, "w");
    tgtCmn = fopen(tgtCmnFile, "w");
    float aPutInit, bPutInit, cPutInit;
    float aPutTgt, bPutTgt, cPutTgt;
    for (loop=0; loop<numCommon; loop++)
    {
        /*
        ** Unfortunately, at this points we must massage the output
        ** so that it conforms to a more precise formatting such
        ** as FORTRAN. If these files are not correctly formatted,
        ** it is possible that the EQUATE program will not work
        ** correctly.
        */
        aPutInit = initCommon[loop][0];
        bPutInit = initCommon[loop][1];
        cPutInit = initCommon[loop][2];

        aPutTgt = tgtCommon[loop][0];
        bPutTgt = tgtCommon[loop][1];
        cPutTgt = tgtCommon[loop][2];
        /* First, deal with the a parameters */
        if (aPutInit < -10)
            fprintf(initCmn, " %f", aPutInit);
        else if (aPutInit < 0 && aPutInit >= -10)
            fprintf(initCmn, " %f", aPutInit);
        else if (aPutInit >= 0 && aPutInit < 10)

```

```

        fprintf(initCmn, " %f", aPutInit);
    else
        fprintf(initCmn, " %f", aPutInit);

    if (aPutTgt < -10)
        fprintf(tgtCmn, " %f", aPutTgt);
    else if (aPutTgt < 0 && aPutTgt >= -10)
        fprintf(tgtCmn, " %f", aPutTgt);
    else if (aPutTgt >= 0 && aPutTgt < 10)
        fprintf(tgtCmn, " %f", aPutTgt);
    else
        fprintf(tgtCmn, " %f", aPutTgt);

    /* and now the b parameters */
    if (bPutInit < -10)
        fprintf(initCmn, " %f", bPutInit);
    else if (bPutInit < 0 && bPutInit >= -10)
        fprintf(initCmn, " %f", bPutInit);
    else if (bPutInit >= 0 && bPutInit < 10)
        fprintf(initCmn, " %f", bPutInit);
    else
        fprintf(initCmn, " %f", bPutInit);

    if (bPutTgt < -10)
        fprintf(tgtCmn, " %f", bPutTgt);
    else if (bPutTgt < 0 && bPutTgt >= -10)
        fprintf(tgtCmn, " %f", bPutTgt);
    else if (bPutTgt >= 0 && bPutTgt < 10)
        fprintf(tgtCmn, " %f", bPutTgt);
    else
        fprintf(tgtCmn, " %f", bPutTgt);

    /* and finally, the c parameters; c should never be negative */
    fprintf(initCmn, " %f\n", cPutInit);
    fprintf(tgtCmn, " %f\n", cPutTgt);

}
fclose(initCmn);
fclose(tgtCmn);

memset(FpointsFile, 0, sizeof(FpointsFile));
strcpy(FpointsFile, Path);
streat(FpointsFile, "fpoints.qd");

```

```

setGPoints(FpointsFile);

Fmax = FVal(initCmnFile, tgtCmnFile, FpointsFile,
            numCommon, GPOINTS, twoPL, threePL);

printf("Calculated Fmax = %f\n", Fmax);

/*
** For the linking part of the study we need to now run the
** EQUATE program. This will be accomplished by first creating
** an input file which can be piped into EQUATE.
**
**          IMPORTANT NOTE: In order for this routine to work properly
**          the output from equate must be suppressed from going to the
**          the printer. This can be accomplished by setting the LPT1 port
**          to a specified file which I can read in later.
*/
memset(runFile, 0, sizeof(runFile));
strcpy(runFile, cmnPath);                                /* put where equate prog. resides */
strcat(runFile, "irtEq.t.run");

memset(newCommon, 0, sizeof(newCommon));
strcpy(newCommon, Path);
strcat(newCommon, "common.new");

memset(newThetas, 0, sizeof(newThetas));
strcpy(newThetas, Path);
strcat(newThetas, makeFName(SE, Model, Relate, NumCom, i, "qnu"));

makeEquate(runFile,
            initCmnFile,
            tgtCmnFile,
            newCommon,
            frEstFile,
            newThetas,
            numCommon,
            twoPL);

/* section working - comment out while testing */
sprintf(cmdString, "equate <%s> scrap.out", runFile);
system(cmdString);

memset(inEquate, 0, sizeof(inEquate));

```

```

strcpy(inEquate, cmnPath);          /* put where equate prog. resides */
strcat(inEquate, "equate.out");

memset(outEquate,0,sizeof(outEquate));
strcpy(outEquate, Path);
strcat(outEquate, "coeffs.out");

memset(acoefStr, 0, sizeof(acoefStr));
memset(kcoefStr, 0, sizeof(kcoefStr));
memset(fcoefStr, 0, sizeof(fcoefStr));
stripEquate(inEquate, outEquate, acoefStr, kcoefStr, fcoefStr);

acoef = atof(acoefStr);
kcoef = atof(kcoefStr);
fcoef = atof(fcoefStr);

printf("F value returned from EQUATE is: %f\n", fcoef);
/*
** Now that we have the equating coefficients and fconverged values
** we can compute the new item estimates for all 50 items. We can
** also compute the Improvement Ratio with this value of f and the Fmax
** value that was computed before.
*/
/*
** First, we must check the 2PL case and rewrite the common item
** file if necessary. This is because EQUATE does not write out
** any values for the c estimates, and the rest of our calculations
** will assume that something is in the c placeholder within the
** file.
*/
if (twoPL)
{
    memset(aNew,0,sizeof(aNew));
    memset(bNew,0,sizeof(bNew));
    /* open existing file for reading in */
    cmnPtr = fopen(newCommon, "r");
    for (loop=0; loop<numCommon; loop++)
    {
        fscanf(cmnPtr, "%f%f", &aNew[loop], &bNew[loop]);
    }
    fclose(cmnPtr);
    /* overwrite same file with 0's added */
    cmnPtr = fopen(newCommon, "w");

```

```

        for (loop=0; loop<numCommon; loop++)
        {
            fprintf(cmnPtr, "%f%f 0.0\n", aNew[loop], bNew[loop]);
        }
        fclose(cmnPtr);
    }

    Fconv = FVal(newCommon, tgtCmnFile, FpointsFile,
                numCommon, GPOINTS, twoPL, threePL);

    printf("My computed fconverge is: %f\n", Fconv);
    if (Fmax != 0)
        ImpRatio = fcoeff/Fmax;
    else
        ImpRatio = 1.0;

    printf("Improvement Ratio is: %lf\n", ImpRatio);
    fprintf(fFile, "%f\t%f\t%lf\n", Fmax, fcoeff, ImpRatio);

    /*
    ** For the equating portion of the study, we need to estimate
    ** the true scores for the various forms (initial, target and
    ** transformed). We also need to compute the real true score
    ** based upon the item and ability parameter files for RMSE
    ** calculations.
    */

    memset(frTrueFile, 0, sizeof(frTrueFile));
    strcpy(frTrueFile, Path);
    strcat(frTrueFile, makeFName(SE,Model,Relate,NumCom,i,"tfr"));
    trueScores(initFile,
               frEstFile,
               frTrueFile,
               numItems,
               numExaminees,
               twoPL,
               threePL);

    memset(toTrueFile, 0, sizeof(toTrueFile));
    strcpy(toTrueFile, Path);
    strcat(toTrueFile, makeFName(SE,Model,Relate,NumCom,i,"tto"));
    trueScores(tgtFile,

```

```

        toEstFile,
        toTrueFile,
        numItems,
        numExaminees,
        twoPL,
        threePL);

    /*
    ** We need to apply the coefficients obtained from the EQUATING
    ** to all of the items for true score estimation.
    */
    memset(newFile, 0, sizeof(newFile));
    strcpy(newFile, Path);
    strcat(newFile, makeFName(SE,Model,Relate,NumCom,i, "new"));

    newPtr = fopen(newFile, "w");
    initPtr = fopen(initFile, "r");

    for (loop=0; loop<numItems; loop++)
    {
        fscanf(initPtr, "%f%f%f", &avalInit, &bvalInit, &cvalInit);
        avalNew = avalInit/acoef;
        bvalNew = bvalInit*acoef + kcoef;
        fprintf(newPtr, "%f\t%f\t%f\n", avalNew, bvalNew, cvalInit);
    }
    fclose(newPtr);
    fclose(initPtr);

    /*
    ** Now, we can compute the estimated true scores for the converted
    ** items and thetas.
    */
    memset(nuTrueFile, 0, sizeof(nuTrueFile));
    strcpy(nuTrueFile, Path);
    strcat(nuTrueFile, makeFName(SE,Model,Relate,NumCom,i,"tnu"));
    trueScores(newFile,
               newThetas,
               nuTrueFile,
               numItems,
               numExaminees,
               twoPL,
               threePL);

```

```

/*
** And last, we want to calculate the true scores for the item
** and theta parameters for RMSE calculations.
*/
if (!trueDone)
{
    memset(prmTrueFile, 0, sizeof(prmTrueFile));
    strcpy(prmTrueFile, Path);
    strcat(prmTrueFile, "trueScor.tpm");
    trueScores(parmFile,
               abilsFile,
               prmTrueFile,
               numItems,
               numExaminees,
               twoPL,
               threePL);

    trueDone = 1;
}

printf("\nValues for true score relationships\n");
seError = stderr(nuTrueFile, prmTrueFile);
printf("Std. Error of converted and parm true scores: %f\n", seError);
fprintf(seTrueFile, "%f ", seError);

seError = stderr(frTrueFile, prmTrueFile);
printf("Std. Error of init and parm true scores: %f\n", seError);
fprintf(seTrueFile, "%f ", seError);

seError = stderr(toTrueFile, prmTrueFile);
printf("Std. Error of target and parm true scores: %f\n", seError);
fprintf(seTrueFile, "%f ", seError);

seError = stderr(frTrueFile, toTrueFile);
printf("Std. Error of init and tgt true scores: %f\n", seError);
fprintf(seTrueFile, "%f ", seError);

seError = stderr(toTrueFile, nuTrueFile);
printf("Std. Error of new and tgt true scores: %f\n", seError);
fprintf(seTrueFile, "%f ", seError);

seError = stderr(frTrueFile, nuTrueFile);

```

```
printf("Std. Error of init and converted true scores: %f\n", seError);
fprintf(seTrueFile, "%f\n", seError);
```

```
printf("\nErrors for theta files\n");
seError = stdErr(newThetas, abilsFile);
printf("Std. Error of converted and parameter : %f\n", seError);
fprintf(seThetaFile, "%f ", seError);
```

```
seError = stdErr(frEstFile, abilsFile);
printf("Std. Error of initial estims and parameters : %f\n", seError);
fprintf(seThetaFile, "%f ", seError);
```

```
seError = stdErr(toEstFile, abilsFile);
printf("Std. Error of target and parameters : %f\n", seError);
fprintf(seThetaFile, "%f ", seError);
```

```
seError = stdErr(frEstFile, toEstFile);
printf("Std. Error of init and target estims: %f\n", seError);
fprintf(seThetaFile, "%f ", seError);
```

```
seError = stdErr(newThetas, toEstFile);
printf("Std. Error of converted and target estims: %f\n", seError);
fprintf(seThetaFile, "%f ", seError);
```

```
seError = stdErr(frEstFile, newThetas);
printf("Std. Error of init and converted estims: %f\n", seError);
```

```
fprintf(seThetaFile, "%f\n", seError);
```

```
} /* end of main for loop */
```

```
fclose(fFile);
fclose(seTrueFile);
fclose(seThetaFile);
```

```
/*
** Very last thing we do is save the random # info
*/
```

```
ranPtr = fopen(ranVals, "w");
fprintf(ranPtr, "%ld\n", iy);
fprintf(ranPtr, "%ld\n", lastSeed);
```

```

    for (i=0; i<NTAB; i++)
        fprintf(ranPtr, "%ld\n", iv[i]);
    fclose(ranPtr);
}

/*
** Put the random number generator routines in here so we can keep
** better track of them.
*/

float ran1(long *idum)
{
    /* my addition */
    FILE *ranPtr;

    int i;
    /* end addition */

    int j;
    long k;
    float temp;

    /* my addition */
    if (newCall) /* first time through this program */
    {
        ranPtr = fopen(ranVals, "r");
        fscanf(ranPtr, "%ld", &iy);
        fscanf(ranPtr, "%ld", idum);
        for (i=0; i<NTAB; i++)
            fscanf(ranPtr, "%ld", &iv[i]);
        newCall = 0;
        fclose(ranPtr);
    }
    /* end addition */

    if (*idum <= 0 || !iy) { /* initialize */
        if (-(*idum) < 1) *idum=1; /* prevent idum = 0 */
        else *idum = -(*idum);
        for (j=NTAB+7; j>=0; j--){ /* load the shuffle table */

```

```

        k=(*idum)/IQ;
        *idum=IA*( *idum-k*IQ)-IR*k;
        if (*idum < 0) *idum += IM;
        if (j < NTAB) iv[j] = *idum;
    }
    iy=iv[0];
}
k=(*idum)/IQ;
*idum=IA*( *idum-k*IQ)-IR*k;
if (*idum < 0) *idum += IM;
j=iy/NDIV;
iy=iv[j];
iv[j] = *idum;
if ((temp=AM*iy) > RNMX) return RNMX;
else return temp;
}

float gasdev(long *idum)
/*
** Returns a normally distributed deviate with zero mean and unit
** variance, using ran1(idum) as the source of uniform deviates
*/
{
    float ran1(long *idum);
    static int iset=0;
    static float gset;
    float fac, rsq, v1, v2;

    if (iset == 0)
    {
        do {
            v1=2.0*ran1(idum)-1.0;
            v2=2.0*ran1(idum)-1.0;
            rsq=v1*v1+v2*v2;
        } while (rsq>= 1.0 || rsq == 0.0);

        fac = sqrt(-2.0*log(rsq)/rsq);

        /*
        ** Now make the Box-Muller transformation to get two normal
        ** deviates. Return one and save the other for next time.
        */
        gset = v1*fac;

```

```
        iset = 1;
        return v2*fac;
    } else {
        iset = 0;
        return gset;
    }
}

float logNormal(long * idum)
{
    float d2, c, tempval;
    float wantMean;
    float wantVar;

    wantMean = MEANA;
    wantVar = VARA;

    d2 = log(1+ wantVar/wantMean);
    c = log(wantMean) - d2/2;
    tempval = c + sqrt(d2)*gasdev(idum);
    return(exp(tempval));
}
```

```

/*
**      File Name:      makeparm.cpp
**      Created:       01/98
**      Author:        Gary Kaskowitz
**                   Department of Measurement, Statistics
**                   and Evaluation
**                   University of Maryland
**
**      Description:   This routine creates non-related error condition
**                   parameters.
*/

```

```
#include "defs.h"
```

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <string.h>
```

```
void createParms(int related, char *parmFile, long *lastSeed, int twoPL)
{
    FILE *myout;

    float outa[numItems];
    float outb[numItems];
    float outc[numItems];
    float coutput;
    long in;
    long in2, in3;
    int i,j;
    char nameit[80];
    float item[numItems][3];
    /* keep item params together */

    /* initialize memory */
    memset(nameit,0,sizeof(nameit));
    memset(outa,0,sizeof(outa));
    memset(outb,0,sizeof(outb));
    memset(outc,0,sizeof(outc));

    myout = fopen(parmFile, "w");

```

```

j=1;
if (!related)
{
    for (i=0; i<numItems; i++)
    {
        outc[i] = ran1(lastSeed);
        outb[i] = gasdev(lastSeed);
        /*
        ** The following piece of code sets a floor or ceiling
        ** whenever we exceed a boundary. It is possible for
        ** this to produce an excessive amount of floor and
        ** ceiling values. A better way is to sample from the
        ** distribution again if we are not in bounds.
        **
        **                               12/14/97
        */
        outa[i] = logNormal(&lastSeed);
        if (outa[i] < minA)
            outa[i] = minA;
        if (outa[i] > maxA)
            outa[i] = maxA;
        /*
        /*
        do {
            outa[i] = logNormal(&lastSeed);
        } while (outa[i] < minA || outa[i] > maxA);
        /*
        outa[i] = logNormal(lastSeed);

        coutput = outc[i]/4;
        if (twoPL)
            coutput = 0.0;
        fprintf(myout, "%f\t%f\t%f\n", outa[i], outb[i], coutput);
    }
}
else
{
    for (i=0; i<numItems; i++)
    {
        outb[i] = gasdev(&in);
        /*

```

```

** Check for whether the value of b is high or low. If it is, we
** want to fiddle with the a and c parms so that they are related
** to the b value received.
*/
if (outb[i] <= LOWB || outb[i] >= HIGHB) /* extreme b value */
{
    do {
        outa[i] = logNormal(lastSeed);
    } while (!(outa[i] <= LOWA /*&& outa[i] >= minA*/)); /*
we want a lower discrim value */

    if (outb[i] >= HIGHB) /* we have a high b value => hard
item */
    {
        /*
        do {
            outa[i] = logNormal(&in3);
        } while (outa[i] >= LOWA);
        */
        do {
            outc[i] = ranl(lastSeed);
        } while (outc[i]/4 <= HIGHC); /*
looking for high guessing */
    }
    else if (outb[i] <= LOWB) /* we have a low b value =>
easy item */
    {
        /*
        do {
            outa[i] = logNormal(&in3);
        } while (outa[i] <= HIGHA);
        */

        do {
            outc[i] = ranl(lastSeed);
        } while (outc[i]/4 >= LOWC); /*
looking for low guessing */
    }
}
else /* we are in middle range */
{
    /*
    do {

```

```

        outa[i] = logNormal(&in3);
    } while (outa[i] <= LOWA || outa[i] >= HIGHA);
    /*
    do {
        outa[i] = logNormal(lastSeed);
    } while (outa[i] < LOWA /*|| outa[i] >= maxA*/);

    do {
        outc[i] = ran1(lastSeed);
    } while (outc[i]/4 <= LOWC || outc[i]/4 >= HIGHC);
    }
    coutput = outc[i]/4;
    if (twoPL)
        coutput = 0.0;
    fprintf(myout, "%f\t%f\t%f\n", outa[i], outb[i], coutput);
} /* end for loop */
} /* end related else */

/*
** Build item structure to keep components together
*/
for (i=0; i<numItems; i++)
{
    item[i][0] = outa[i];
    item[i][1] = outb[i];
    item[i][2] = outc[i]/4;
    if (twoPL)
        item[i][2] = 0.0;
}
fclose(myout);
}

```

```

/*
**      File Name:          related.cpp
**      Created:           01/98
**      Author:            Gary Kaskowitz
**                        Department of Measurement, Statistics
**                        and Evaluation
**                        University of Maryland
**
**      Description:       This routine generates related item parameters
**                        in accordance with methodology discussion.
*/

```

```

#include <stdio.h>
#include <math.h>

```

```

#include "defs.h"

```

```

void genRelate(int level,
               char *aRelatedFile,
               char *bRelatedFile,
               char *itemParamFile,
               char *initFile,
               float retItems[numItems][3],
               long *lastSeed,
               int twoPL)
{
    extern float gasdev(long *);
    FILE *mya, *myb, *myItems, *initEst, *tgtEst;

    float Lowa[5], Lowb[5];
    float Meda[5], Medb[5];
    float Hia[5], Hib[5];
    float aval[numItems], bval[numItems], cval[numItems];
    float aEstInit[numItems], bEstInit[numItems], cEstInit[numItems];
    float aEstTgt[numItems], bEstTgt[numItems], cEstTgt[numItems];

    int i, a_posn, b_posn;
    float bottom_a, low_mid_a, up_mid_a, upper_a;
    /*
    ** Determine the type of error we are generating (function
    ** and level).
    */

```

```

mya = fopen(aRelatedFile, "r");
myb = fopen(bRelatedFile, "r");

for (i=0; i<5; i++)
{
    fscanff(mya, "%f", &Lowa[i]);
    fscanff(myb, "%f", &Lowb[i]);
}
for (i=0; i<5; i++)
{
    fscanff(mya, "%f", &Meda[i]);
    fscanff(myb, "%f", &Medb[i]);
}
for (i=0; i<5; i++)
{
    fscanff(mya, "%f", &Hia[i]);
    fscanff(myb, "%f", &Hib[i]);
}
fclose(mya);
fclose(myb);

/*
** Read in item parameter estimates
*/
myItems = fopen(itemParamFile, "r");
for (i=0; i<numItems; i++)
{
    fscanff(myItems, "%f%f%f", &aval[i], &bval[i], &cval[i]);
}
fclose(myItems);

/*
** Now choose the appropriate level and start generating
** items to correspond to the specified level.
*/
for (i=0; i<numItems; i++)
{
    /*
    ** break the b scale into 5 segments for proper assignment
    */
    if (bval[i] <= -2.25)
        b_posn = 0;
}

```

```

else if (bval[i] > -2.25 && bval[i] <= -.75)
    b_posn = 1;
else if (bval[i] > -.75 && bval[i] <= .75)
    b_posn = 2;
else if (bval[i] > .75 && bval[i] <= 2.25)
    b_posn = 3;
else if (bval[i] > 2.25)
    b_posn = 4;
/*
** and do the same thing for a errors
*/
bottom_a = MEANA - .25; /* one deviation below mean */
low_mid_a = MEANA;
up_mid_a = MEANA + .25; /* one deviation above mean */
upper_a = MEANA + .5; /* two deviations above mean */
if (aval[i] <= bottom_a)
    a_posn = 0;
else if (aval[i] > bottom_a && aval[i] <= low_mid_a)
    a_posn = 1;
else if (aval[i] > low_mid_a && aval[i] <= up_mid_a)
    a_posn = 2;
else if (aval[i] > up_mid_a && aval[i] <= upper_a)
    a_posn = 3;
else if (aval[i] > upper_a)
    a_posn = 4;

switch (level){
case 1:
    /*
    ** Put a floor and ceiling on the values of a so that
    ** it is within the range .8 to 2.5 (minA and maxA)
    */
    aEstInit[i] = gasdev(lastSeed)*Lowa[a_posn] + aval[i];
/*
    if (aest[i] < minA)
        aest[i] = minA;
    else if (aest[i] > maxA)
        aest[i] = maxA;
*/

    bEstInit[i] = gasdev(lastSeed)*Lowb[b_posn] + bval[i];
    cEstInit[i] = gasdev(lastSeed)*LOWCERR + cval[i];

    /* make sure that c is within 0-1 bounds */

```

```

        if (cEstInit[i] < 0)
            cEstInit[i] = 0;
        else if (cEstInit[i] > 1)
            cEstInit[i] = 1;
        break;
    case 2:
/*
        do {
            aest[i] = gasdev(lastSeed)*Meda[a_posn] + aval[i];
        } while (aest[i] < minA || aest[i] > maxA);
*/
        aEstInit[i] = gasdev(lastSeed)*Meda[a_posn] + aval[i];
        bEstInit[i] = gasdev(lastSeed)*Medb[b_posn] + bval[i];
        cEstInit[i] = gasdev(lastSeed)*MEDCERR + cval[i];

        if (cEstInit[i] < 0)
            cEstInit[i] = 0;
        else if (cEstInit[i] > 1)
            cEstInit[i] = 1;

        break;
    case 3:
/*
        do {
            aest[i] = gasdev(lastSeed)*Hia[a_posn] + aval[i];
        } while (aest[i] < minA || aest[i] > maxA);
*/
        aEstInit[i] = gasdev(lastSeed)*Hia[a_posn] + aval[i];
        bEstInit[i] = gasdev(lastSeed)*Hib[b_posn] + bval[i];
        cEstInit[i] = gasdev(lastSeed)*HICERR + cval[i];
        if (cEstInit[i] < 0) cEstInit[i] = 0;
        if (cEstInit[i] > 1) cEstInit[i] = 1;

        break;
    }
}

initEst = fopen(initFile, "w");
for (i=0; i<numItems; i++)
{
    /* check for twoPL case and 0 out c if so */
    if (twoPL)
        cEstInit[i] = 0.0;
}

```

```
fprintf(initEst, "%f\t%f\t%f\t\n", aEstInit[i], bEstInit[i], cEstInit[i]);  
retItems[i][0] = aEstInit[i];  
retItems[i][1] = bEstInit[i];  
retItems[i][2] = cEstInit[i];  
}  
fclose(initEst);  
return;  
}
```

```

/*
**      File Name:      qanswers.cpp
**      Created:      01/98
**      Author:      Gary Kaskowitz
**                  Department of Measurement, Statistics
**                  and Evaluation
**                  University of Maryland
**
**      Description:   This routine creates an answer file for a file
**                  of ability estimates and appropriate IRT model.
*/

```

```
#include "defs.h"
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```

void genAnswers(int twoPL, int threePL,
                char *parmFile,
                char *abilFile,
                char *ansFile,
                long *lastSeed)
{
    long in;
    FILE *myItems, *myAnswers, *outThetas;
    int i,j;
    float theta[numExaminees];
    float aval[numItems], bval[numItems], cval[numItems];
    double top, bottom, pVal;
    int correct;
    int answer[numExaminees][numItems];

    /*
    ** Read parameters file into [3][50] array for processing.
    */
    myItems = fopen(parmFile, "r");
    for (i=0; i<numItems; i++)
    {
        fscanf(myItems, "%f%f%f", &aval[i], &bval[i], &cval[i]);
    }
    fclose(myItems);

```

```

/*
** Generate theta array and write to file for later processing
*/
outThetas = fopen(abilFile, "w");
for (j=0; j<numExaminees; j++)
{
    theta[j] = gasdev(lastSeed);
    fprintf(outThetas, "%f\n", theta[j]);
}
fclose(outThetas);

/*
** Loop through all items for each theta. Calculate prob.
** of correct response. Compare this value against random
** Uniform[0,1] number for correct response choice (i.e.,
** if P is greater than ran, put a 1 in the response vector
*/
for (j=0; j<numExaminees; j++)
{
    for (i=0; i<numItems; i++)
    {
        top = exp(aval[i]*(theta[j] - bval[i]));
        bottom = 1 + top;
        if (twoPL)
        {
            pVal = top/bottom;
        }
        else if (threePL)
            pVal = cval[i] + (1-cval[i])*(top/bottom);

        if (pVal >= ran1(lastSeed))
            correct = 1;
        else
            correct = 0;

        answer[j][i] = correct;
    }
}

/*
** Write out the answer array to a file for future processing

```

```
*/  
  
/* name the file appropriately */  
  
myAnswers = fopen(ansFile, "w");  
for (j=0; j<numExaminees; j++)  
{  
    for (i=0; i<numItems; i++)  
    {  
        fprintf(myAnswers, "%d ", answer[j][i]);  
    }  
    fprintf(myAnswers, "\n");  
}  
fclose(myAnswers);  
}
```

```

/*
**      File Name:      Weights.cpp
**      Created:       01/98
**      Author:        Gary Kaskowitz
**                   Department of Measurement, Statistics
**                   and Evaluation
**                   University of Maryland
**
**      Description:   This program computes quadrature weights
**                   for EAP estimation.
*/

```

```
#include "defs.h"
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```
float prob[numQuads]={0};
```

```
float weight[numQuads];
```

```
void getWeights(float quadLow, float quadHigh, char *weightFile)
```

```
{
```

```
    extern floatprob[numQuads];
```

```
    int i, intervals;
```

```
    float q;
```

```
    float quadRange, val, spacing;
```

```
    float smallAmt = .0001;
```

```
    FILE *myPtr;
```

```
    float sum=0;
```

```
    /*
```

```
    ** Determine the quadrature points that we will be estimating
    ** weights for.
```

```
    */
```

```
    quadRange = quadHigh - quadLow;
```

```
    intervals = numQuads - 1;
```

```
    spacing = quadRange/intervals;
```

```
/*
** Calculate the discrete prior probabilities at each of these points
*/
i=0;
for (q=quadLow; q<=quadHigh+smallAmt; q+=spacing)
{
    val = (1/sqrt(2*pi))*exp(-(q*q)/2);
    prob[i] = val;
    i++;
}
/*
** Adjust the probabilities by the differences between successive points
*/
myPtr = fopen(weightFile, "w");
for (i=0; i<numQuads; i++)
{
    weight[i] = prob[i] * spacing;
    fprintf(myPtr, "%f\n", weight[i]);
}
fclose(myPtr);
}
```

```

/*
**      File Name:      genitems.cpp
**      Created:       01/98
**      Author:        Gary Kaskowitz
**                   Department of Measurement, Statistics
**                   and Evaluation
**                   University of Maryland
**
**      Description:   This routine generates items for a form based
**                   upon level of error, model type and item
**                   parameters.
*/

```

```

#include <stdio.h>
#include <math.h>
#include <memory.h>

```

```

#include "defs.h"

```

```

/*
** genItems: This routine generates the item parameter
**           estimates necessary for initial and target
**           forms.
**
**           01/09/98
*/

```

```

void genItems(int level,
              int twoPL,
              char *itemParamFile,
              char *initFile,
              float retItems[numItems][3],
              long *lastSeed)
{
    extern float gasdev(long *);
    FILE *myItems, *initEst, *tgtEst;

    float aval[numItems], bval[numItems], cval[numItems];
    float aEstInit[numItems], bEstInit[numItems], cEstInit[numItems];
    float aEstTgt[numItems], bEstTgt[numItems], cEstTgt[numItems];

    int i;

    /*

```

```

** Eliminate likelihood of writing junk (especially with 2PL case)
*/
memset(aEstInit,0,sizeof(aEstInit));
memset(bEstInit,0,sizeof(bEstInit));
memset(cEstInit,0,sizeof(cEstInit));

/*
** Read in item parameter estimates
*/
myItems = fopen(itemParamFile, "r");
for (i=0; i<numItems; i++)
{
    fscanf(myItems, "%f%f%f", &aval[i], &bval[i], &cval[i]);
}
fclose(myItems);

/*
** Now choose the appropriate level and start generating
** items to correspond to the specified level.
*/
for (i=0; i<numItems; i++)
{
    switch (level){
    case 1:
        if (twoPL)
        {
            /* first, select the initial form estimate */
            aEstInit[i] = gasdev(lastSeed)*LOWA2 + aval[i];
            bEstInit[i] = gasdev(lastSeed)*LOWB2 + bval[i];
            /* put in a placeholder for c */
            cEstInit[i] = 0.0;
        }
        else
        {
            aEstInit[i] = gasdev(lastSeed)*LOWA3 + aval[i];
            bEstInit[i] = gasdev(lastSeed)*LOWB3 + bval[i];
            cEstInit[i] = gasdev(lastSeed)*LOWCERR + cval[i];

            /* check c for floor and ceiling */
            if (cEstInit[i] < 0) cEstInit[i] = 0;
            if (cEstInit[i] > .25) cEstInit[i] = .25;
        }
    }
}

```

```

        break;
    case 2:
        if (twoPL)
        {
            aEstInit[i] = gasdev(lastSeed)*MODA2 + aval[i];
            bEstInit[i] = gasdev(lastSeed)*MODB2 + bval[i];
            /* put in a placeholder for c */
            cEstInit[i] = 0.0;
        }
        else
        {
            aEstInit[i] = gasdev(lastSeed)*MODA3 + aval[i];
            bEstInit[i] = gasdev(lastSeed)*MODB3 + bval[i];
            cEstInit[i] = gasdev(lastSeed)*MEDCERR + cval[i];

            /* check c for floor and ceiling */
            if (cEstInit[i] < 0) cEstInit[i] = 0;
            if (cEstInit[i] > .25) cEstInit[i] = .25;
        }
        break;
    case 3:
        if (twoPL)
        {
            aEstInit[i] = gasdev(lastSeed)*HIA2 + aval[i];
            bEstInit[i] = gasdev(lastSeed)*HIB2 + bval[i];
            /* put in a placeholder for c */
            cEstInit[i] = 0.0;
        }
        else
        {
            aEstInit[i] = gasdev(lastSeed)*HIA3 + aval[i];
            bEstInit[i] = gasdev(lastSeed)*HIB3 + bval[i];
            cEstInit[i] = gasdev(lastSeed)*HICERR + cval[i];

            /* check c for floor and ceiling */
            if (cEstInit[i] < 0) cEstInit[i] = 0;
            if (cEstInit[i] > .25) cEstInit[i] = .25;
        }
        break;
    }
}

initEst = fopen(initFile, "w");

```

```
for (i=0; i<numItems; i++)
{
    fprintf(initEst, "%f\t%f\t%f\t\n", aEstInit[i], bEstInit[i], cEstInit[i]);
    retItems[i][0] = aEstInit[i];
    retItems[i][1] = bEstInit[i];
    retItems[i][2] = cEstInit[i];
}
fclose(initEst);
return;
}
```

```

/*
**      File Name:          calcQuad.cpp
**      Created:           01/98
**      Author:            Gary Kaskowitz
**                        Department of Measurement, Statistics
**                        and Evaluation
**                        University of Maryland
**
**      Description:       This routine calculates evenly space quadrature
**                        points based upon number of points and theta
**                        range.
*/

#include "defs.h"

#include <math.h>
#include <stdio.h>

void calcQuadPts(float quadLow, float quadHigh, int thetaPts, char *quadFile)
{
    float quadRange, intervals, spacing, k;
    FILE *quadPts;
    float smallAmt = .0001;

    quadRange = quadHigh - quadLow;
    intervals = thetaPts - 1;

    spacing = quadRange/intervals;

    quadPts = fopen(quadFile, "w");
    for (k=quadLow; k<=quadHigh+smallAmt; k+=spacing)
        fprintf(quadPts, "%f\n", k);
    fclose(quadPts);
}

```

```

/*
**      File Name:      eap.cpp
**      Created:      01/98
**      Author:      Gary Kaskowitz
**                  Department of Measurement, Statistics
**                  and Evaluation
**                  University of Maryland
**
**      Description:   This routine performs EAP estimation of abilities
**                  based upon item estimates, answers and model type.
*/

```

```
#include "defs.h"
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```
#include <memory.h>
```

```
void eap(char *itemEstFile,
         char *wgtFile,
         char *quadFile,
         char *answerFile,
         int twoPL, int threePL,
         char *estimsFile)

```

```
{
```

```
    FILE *myItems;
```

```
    FILE *myWgts;
```

```
    FILE *myQuads;
```

```
    FILE *myAnswer;
```

```
    FILE *myEstims;
```

```
    double probRight, probWrong;
```

```
    double top,bottom;
```

```
    double totalL;
```

```
    float aval[numItems],bval[numItems],cval[numItems];
```

```
    float quadPt[numQuads], weight[numQuads];
```

```
    int answer[numExaminees][numItems];
```

```
    double topEst=0,bottomEst=0;
```

```
    float estTheta[numExaminees];
```

```
    int i,j,k;
```

```

/* initialize appropriate memory */
memset(estTheta, 0, sizeof(estTheta));

/*
** Read in item parameter estimates
*/
myItems = fopen(itemEstFile, "r");
for (i=0; i<numItems; i++)
{
    fscanf(myItems, "%f%f%f", &aval[i], &bval[i], &cval[i]);
}
fclose(myItems);
/*
** Establish quadrature points and prior weights for each
** quadrature point.
*/
myWgts = fopen(wgtFile, "r");
myQuads = fopen(quadFile, "r");
for (k=0; k<numQuads; k++)
{
    fscanf(myWgts, "%f", &weight[k]);
    fscanf(myQuads, "%f", &quadPt[k]);
}
fclose(myWgts);
fclose(myQuads);
/*
** Open the answer file
*/
myAnswer = fopen(answerFile, "r");
for (j=0; j<numExaminees; j++)
    for (i=0; i<numItems; i++)
        fscanf(myAnswer, "%d", &answer[j][i]);
fclose(myAnswer);

/*
** Loop through each respondent
*/
for (j=0; j<numExaminees; j++)
{
    /*
    ** Read in response vector file (on the PC, we will do this one line
    ** at a time. On the mainframe, it makes more sense to read the entire
    ** thing into memory. If we try to read it all into memory at once on

```

```

** the PC, we will run into problems with highmem, etc. when we attempt
** to malloc our memory).
*/

```

```

/*
** Loop through each quadrature point
*/
for (k=0; k<numQuads; k++)

```

```

{
    /* initialize the L value */

```

```

    totalL = 1;

```

```

    /*

```

```

    ** Loop through each item in the response vector

```

```

    */

```

```

    for (i=0; i<numItems; i++)

```

```

    {

```

```

        /* calculate P(theta) and 1 - P(theta) for each quad pt. */
        top = exp(aval[i]*(quadPt[k]-bval[i]));
        bottom = 1 + top;

```

```

        if (twoPL)

```

```

            probRight = top/bottom;

```

```

        else if (threePL)

```

```

            probRight = cval[i] + (1-cval[i])*(top/bottom);

```

```

        probWrong = 1 - probRight;

```

```

        if (answer[j][i] == 1)

```

```

            /* we have a correct

```

```

            totalL = totalL * probRight;

```

```

        else if (answer[j][i] == 0) /* we have an incorrect

```

```

            totalL = totalL * probWrong;

```

```

        }

```

```

    /*

```

```

    ** We now have our L value for this particular quad point.

```

```

    ** our values to estimate theta for this particular person

```

```

    /*
    topEst = topEst + quadPt[k]*totalL*weight[k];
    bottomEst = bottomEst + totalL*weight[k];

```

```

}
/*

```

```

** We can now calculate our best estimate of theta for person j

```

response */

response */

Update

```
        */
        estTheta[j] = topEst/bottomEst;
        /* Reset the estimates */
        topEst = 0;
        bottomEst = 0;
    }
    myEstims = fopen(estimsFile, "w");
    for (j=0; j<numExaminees; j++)
        if (estTheta[j] < 0) /* negative, put in one less space */
            fprintf(myEstims, "%f\n", estTheta[j]);
        else
            fprintf(myEstims, " %f\n", estTheta[j]);
    fclose(myEstims);
}
```

```

/*                               Utilities.cpp
**   File Name:                   01/98
**   Created:                     Gary Kaskowitz
**   Author:                      Department of Measurement, Statistics
**                               and Evaluation
**                               University of Maryland
**
**   Description:  Various utility functions such as make_file_name
**               picking common items, sorting common items, etc.
*/

```

```

#include "defs.h"
#include <stdio.h>
#include <memory.h>
#include <math.h>
#include <string.h>

```

```

char *makeFName(char SE, char PL, char rel, char *items, int count, char *ext)
{
    char outname[80];

    memset(outname,0,sizeof(outname));

    if (count < 10)
        sprintf(outname, "%c%c%c%c%s00%d.%s", SE,PL,rel,items,count,ext);
    else if (count < 100 && count >=10)
        sprintf(outname, "%c%c%c%c%s0%d.%s", SE,PL,rel,items,count,ext);
    else
        sprintf(outname, "%c%c%c%c%s%d.%s", SE,PL,rel,items,count,ext);

    return(outname);
}

```

```

void pickCommon(float items[numItems][3], int numCommon, float
common[maxCommon][3])
{
    int i,j;
    int increment;
    int flipFlop = 0;

```

```
/*
** Choose the appropriate increment for the common items.
*/
switch (numCommon)
{
case 5:
    increment = 12;
    break;
case 15:
    increment = 3;
    break;
case 25:
    increment = 2;
    break;
}

/*
** Now choose the appropriate items from the array of bvals
*/
memset(common, 0, sizeof(common));
for (i=0,j=0; i<numItems; i+=increment,j++)
{
    /*
    ** In the case of 15 common items, we need to alternate
    ** the increment between 3 and 4 in order to cover the
    ** full range of the spectrum (i.e., 50 items).
    */
    if (numCommon == 15)
    {
        if (flipFlop)
        {
            i++;
            flipFlop = 0;
        }
        else
            flipFlop = 1;
    }

    common[j][0] = items[i][0];
    common[j][1] = items[i][1];
    common[j][2] = items[i][2];
}
return;
```

```

}

/*
** Following algorithm modified from numerical recipes
**
**          12/97 - GSK
**
*/

void piksrt2(int n,
             float sortArr[numItems][3],
             float tgtArr[numItems][3])
{
    int i,j;
    float aSort,bSort,cSort;
    float aTgt, bTgt, cTgt;

    for (j=1; j<n; j++)
    {
        aSort=sortArr[j][0];
        bSort=sortArr[j][1];
        cSort=sortArr[j][2];

        aTgt = tgtArr[j][0];
        bTgt = tgtArr[j][1];
        cTgt = tgtArr[j][2];

        i=j-1;
        while (i>=0 && sortArr[i][1] > bSort)
        {
            sortArr[i+1][0] = sortArr[i][0];
            sortArr[i+1][1] = sortArr[i][1];
            sortArr[i+1][2] = sortArr[i][2];

            tgtArr[i+1][0] = tgtArr[i][0];
            tgtArr[i+1][1] = tgtArr[i][1];
            tgtArr[i+1][2] = tgtArr[i][2];

            i--;
        }
        sortArr[i+1][0] = aSort;
        sortArr[i+1][1] = bSort;
    }
}

```

```

        sortArr[i+1][2] = cSort;

        tgtArr[i+1][0] = aTgt;
        tgtArr[i+1][1] = bTgt;
        tgtArr[i+1][2] = cTgt;
    }
}

/*
** probRight() performs the basic IRT function to determine the probability
** of a correct response to an item given a particular theta and appropriate
** item parameters. This function is a general utility which is used
** throughout the application.
**
** 12/19/97 - gsk
*/

double probRight(float aval,
                 float bval,
                 float cval,
                 float theta,
                 int twoPL,
                 int threePL)
{
    double pVal = 0;
    double top = 0;
    double bottom = 0;

    top = exp(aval*(theta - bval));
    bottom = 1 + top;
    if (twoPL)
        pVal = top/bottom;
    else if (threePL)
        pVal = cval + (1-cval)*(top/bottom);

    return(pVal);
}

/*
** Calculate the square of a number
*/
double square(double value)

```

```

{
    return(value*value);
}

void setGPoints(char *fname)
{
    calcQuadPts(-4.0, 4.0, GPOINTS, fname);
}

double expectedScore(float aval[MAXITEMS],
                    float bval[MAXITEMS],
                    float cval[MAXITEMS],
                    int itemRange,
                    float theta,
                    int twoPL,
                    int threePL)
{
    FILE *myItems;
    double totProb=0;
    int i;

    for (i=0; i<itemRange; i++)
    {
        totProb += probRight(aval[i],bval[i],cval[i],theta,twoPL,threePL);
    }
    return(totProb);
}

double FVal(char *initEstFile,
            char *tgtEstFile,
            char *thetaPtFile,
            int itemRange,
            int numThetas,
            int twoPL,
            int threePL)
{
    FILE *myItems;
    FILE *myThetas;
    float avalInit[MAXITEMS], bvalInit[MAXITEMS], cvalInit[MAXITEMS];
    float avalTgt[MAXITEMS], bvalTgt[MAXITEMS], cvalTgt[MAXITEMS];
    float estPt[MAXTHETAS];

```

```

double sumScore=0;
double partInit=0, partTgt=0, sumDiff=0, fvalue=0;
int i;

/*
** Initialize everything
*/
memset(estPt,0,sizeof(estPt));
memset(avalInit,0,sizeof(avalInit));
memset(bvalInit,0,sizeof(bvalInit));
memset(cvalInit,0,sizeof(bvalInit));

memset(avalTgt,0,sizeof(avalTgt));
memset(bvalTgt,0,sizeof(bvalTgt));
memset(cvalTgt,0,sizeof(bvalTgt));
/*
** Read in item parameter estimates for initial form
*/
myItems = fopen(initEstFile, "r");
for (i=0; i<itemRange; i++)
{
    fscanf(myItems, "%f%f%f", &avalInit[i], &bvalInit[i], &cvalInit[i]);
}
fclose(myItems);

/*
** Read in item parameter estimates for target form
*/
myItems = fopen(tgtEstFile, "r");
for (i=0; i<itemRange; i++)
{
    fscanf(myItems, "%f%f%f", &avalTgt[i], &bvalTgt[i], &cvalTgt[i]);
}
fclose(myItems);

/*
** Read in theta points which scores are being estimated for
*/
myThetas = fopen(thetaPtFile, "r");
for (i=0; i<numThetas; i++)
{
    fscanf(myThetas, "%f", &estPt[i]);
}

```

```

fclose(myThetas);

for (i=0; i<numThetas; i++)
{
    partInit = expectedScore(avalInit,bvalInit,cvalInit,itemRange,
                             estPt[i], twoPL,
threePL);

    partTgt = expectedScore(avalTgt,bvalTgt,cvalTgt,itemRange,
                             estPt[i], twoPL,
threePL);

    sumDiff += square(partInit-partTgt);

}
fvalue = sumDiff/numThetas;

return(fvalue);
}

void trueScores(char *itemEstFile,
                char *thetaPtFile,
                char *truePtFile,
                int itemRange,
                int numThetas,
                int twoPL,
                int threePL)
{
    FILE *myItems;
    FILE *myThetas;
    FILE *myTrue;
    float aval[MAXITEMS], bval[MAXITEMS], cval[MAXITEMS];
    float estPt[MAXTHETAS];
    double trueScore=0;
    int i;

    /*
    ** Initialize everything
    */
    memset(estPt,0,sizeof(estPt));

```

```

    memset(aval,0,sizeof(aval));
    memset(bval,0,sizeof(bval));
    memset(cval,0,sizeof(bval));
    /*
    ** Read in item parameter estimates
    */
    myItems = fopen(itemEstFile, "r");
    for (i=0; i<numItems; i++)
    {
        fscanf(myItems, "%f %f %f", &aval[i], &bval[i], &cval[i]);
    }
    fclose(myItems);

    /*
    ** Read in theta points which scores are being estimated for
    */
    myThetas = fopen(thetaPtFile, "r");
    for (i=0; i<numThetas; i++)
    {
        fscanf(myThetas, "%f", &estPt[i]);
    }
    fclose(myThetas);

    /*
    ** Prep the true score output file for values
    */
    myTrue = fopen(truePtFile, "w");

    for (i=0; i<numThetas; i++)
    {
        trueScore = expectedScore(aval,bval,cval,itemRange,
                                estPt[i],twoPL,
threePL);
        fprintf(myTrue, "%f\n", trueScore);
    }
    /* Close true score file */
    fclose(myTrue);
}

```

```

/*****/
/* The following routine creates the batch file which is needed in */
/* order to run the equate program */
/*****/
void makeEquate(char *eqtFile,
                char *initCmnItems,
                char *tgtCmnItems,
                char *outCmnItems,
                char *initThetas,
                char *outThetas,
                int numCommon,
                int twoPL)
{
    FILE *eqtPtr;

    eqtPtr = fopen(eqtFile, "w");
    fprintf(eqtPtr, "%s\n", "SEE OF EQUATING STUDY"); /* name of test run */
    fprintf(eqtPtr, "%d\n", GPOINTS); /* # of scale points */
    fprintf(eqtPtr, "%c\n", 'D'); /*
dichotomous model */
    if (twoPL) /* 2 or
3PL model */
        fprintf(eqtPtr, "%c\n", '2');
    else
        fprintf(eqtPtr, "%c\n", '3');
    fprintf(eqtPtr, "%s\n", initCmnItems); /* from items file */
    if (twoPL) /*
format of file */
        fprintf(eqtPtr, "%s\n", "(2F12.6)");
    else
        fprintf(eqtPtr, "%s\n", "(3F12.6)");
    fprintf(eqtPtr, "%d\n", numCommon); /* # of common items
*/
    fprintf(eqtPtr, "%c\n", 'L'); /* logistic
model */
    fprintf(eqtPtr, "%s\n", tgtCmnItems); /* target items file */
    if (twoPL) /*
format of file */
        fprintf(eqtPtr, "(2F12.6)\n");
    else
        fprintf(eqtPtr, "(3F12.6)\n");
    fprintf(eqtPtr, "%d\n", numCommon); /* # of common items

```

```

*/
model   fprintf(eqtPtr, "%c\n", 'L');
        /*
        fprintf(eqtPtr, "%s\n", outCmnItems);
        fprintf(eqtPtr, "%c\n", 'Y');
*/
        fprintf(eqtPtr, "%s\n", initThetas);
        fprintf(eqtPtr, "%s\n", "(F10.6)");
        fprintf(eqtPtr, "%d\n", numExaminees);
        fprintf(eqtPtr, "%s\n", outThetas);
        fprintf(eqtPtr, "%c\n", 'Y');
        fprintf(eqtPtr, "1-%d:\n", numCommon);
        fprintf(eqtPtr, "1-%d:\n", numCommon);
        */

        fclose(eqtPtr);
}

```

```

/* logistic
/* converted items */
/* LOOK UP!
/* initial theta file */
/* theta file format */
/* num. of theta pts */
/* output theta file */
/* formatting */
/* same */

```

```

/*
**      File Name:          stderr.cpp
**      Created:           01/98
**      Author:            Gary Kaskowitz
**                        Department of Measurement, Statistics
**                        and Evaluation
**                        University of Maryland
**
**      Description:       This routine computes the standard error between
**                        the values in two different input files.
*/

```

```
#include "defs.h"
```

```
#include <math.h>
```

```
#include <stdio.h>
```

```

double stdErr(char *abilsFile, char *estimsFile)
{
    FILE *abils, *estims;
    float thetas[numExaminees], thetaHats[numExaminees];
    double diff=0, totalDiff=0, error=0, totError=0;
    int j;

    abils = fopen(abilsFile, "r");
    estims = fopen(estimsFile, "r");

    for (j=0; j<numExaminees; j++)
    {
        fscanf(abils, "%f", &thetas[j]);
        fscanf(estims, "%f", &thetaHats[j]);
    }
    fclose(abils);
    fclose(estims);

    for (j=0; j<numExaminees; j++)
    {
        diff = thetas[j] - thetaHats[j];
        totalDiff = totalDiff + diff*diff;
    }
    totError = totalDiff/numExaminees;
    error = sqrt(totError);
}

```

```
        return(error);
    }

double Dvalue(char *trueFile1, char *trueFile2)
{
    FILE *file1, *file2;
    float true1[numExaminees], true2[numExaminees];
    double diff=0, totalDiff=0, error=0, totError=0;
    int j;

    file1 = fopen(trueFile1, "r");
    file2 = fopen(trueFile2, "r");

    for (j=0; j<numExaminees; j++)
    {
        fscanf(file1, "%f", &true1[j]);
        fscanf(file2, "%f", &true2[j]);
    }
    fclose(file1);
    fclose(file2);

    for (j=0; j<numExaminees; j++)
    {
        diff = true1[j] - true2[j];
        totalDiff = totalDiff + sqrt(diff*diff);
    }
    return(totalDiff);
}
```

```

/*
**      File Name:      stripEquate.cpp
**      Created:      01/98
**      Author:       Gary Kaskowitz
**                  Department of Measurement, Statistics
**                  and Evaluation
**                  University of Maryland
**
**      Description:   This program strips the A, K and F values out
**                  of an EQUATE program output file.
*/

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```

void stripEquate(char *inputFile,
                 char *outputFile,
                 char *aval,
                 char *kval,
                 char *fval)
{

```

```

    char input[80];
    char line1[80];
    char line2[80];
    char junk1[20], junk2[20], junk3[20];
    /*char aval[20], kval[20], fval[20];*/

```

```

    FILE *myPtr, *myOut;
    int notFound = 0;

```

```

    /* initialize strings prior to parsing */
    memset(input,0,sizeof(input));
    memset(line1,0,sizeof(line1));
    memset(line2,0,sizeof(line2));

```

```

    myPtr = fopen(inputFile, "r");
    fgets(input, 80, myPtr);
    while ((strncmp(input, " METRIC TRANS", 13)))
    {

```

```

        if (fgets(input, 80, myPtr)==NULL)
        {

```

```

            notFound = 1;

```

```

        } /* line never found */
    }
}

```

```
                break;
            }
        }
    if (!notFound) /* go ahead and read next two lines */
    {
        fscanf(myPtr, "%s %s %s %s", junk1, aval, junk2, kval);
        fscanf(myPtr, "%s %s %s %s", junk1, junk2, junk3, fval);
    }
    /*
    ** Output A, K and F to our output file for future analysis
    */
    myOut = fopen(outputFile, "a");
    fprintf(myOut, "%s\t%s\t%s\n", aval, kval, fval);
    fclose(myOut);
    fclose(myPtr);
}
```

References

- Baker, F. B. (1984). Ability metric transformations involved in vertical equating under item response theory. Applied Psychological Measurement, 8, 261-271.
- Baker, F. B. (1990). Some observations on the metric of PC-BILOG results. Applied Psychological Measurement, 14, 139-150.
- Baker, F. B. (1992). Equating tests under the graded response model. Applied Psychological Measurement, 16, 87-96.
- Baker, F. B. (1993). Equating tests under the nominal response model. Applied Psychological Measurement, 17, 239-251.
- Baker, F. B. (1996). An investigation of the sampling distributions of equating coefficients. Applied Psychological Measurement, 20, 45-57.
- Baker, F. B. (1997). A note on the proper use of the Numerical Recipes RAN1 random number generator. Computational Statistics & Data Analysis, 25, 237-239.
- Baker, F. B., Al-Karni, A., & Al-Dosary, I. M. (1991). EQUATE: A computer program for the test characteristic curve method of IRT equating. Applied Psychological Measurement, 15, 78.
- Baker, F. B., & Al-Karni, A. (1991). A comparison of two procedures for computing IRT equating coefficients. Journal of Educational Measurement, 28, 147-162.

Bock, R. D. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. Psychometrika, 37, 29-51.

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

Brennan, R. L., & Kolen, M. J. (1987a). A reply to Angoff. Applied Psychological Measurement, 11, 301-306.

Brennan, R. L., & Kolen, M. J. (1987b). Some practical issues in equating. Applied Psychological Measurement, 11, 279-290.

Budescu, D. (1985). Efficiency of linear equating as a function of the length of the anchor test. Journal of Educational Measurement, 22, 13-20.

Camilli, G., Wang, M., & Fesq, J. (1995). The effects of dimensionality on equating the Law School Admission Test. Journal of Educational Measurement, 32, 79-96.

Camilli, G., Yamamoto, K., & Wang, M. (1993). Scale shrinkage in vertical equating. Applied Psychological Measurement, 17, 379-388.

Cook, L. L., & Eignor, D. R. (1991). IRT equating methods. Educational Measurement: Issues and Practices, 37-45.

Cook, L. L., & Petersen, N. S. (1987). Problems related to the use of conventional and item response theory equating methods in less than optimal circumstances. Applied

Psychological Measurement, 11, 225-244.

Davey, T. (1996). Linking multidimensional item calibrations. Applied Psychological Measurement, 20, 405-416.

De Ayala, R. J., Schafer, W. D., & Sava-Bolesta, M. (1995). An investigation of the standard errors of expected a posteriori. British Journal of Mathematical and Statistical Psychology, 48, 385-405.

De Gruijter, D. N. M. (1984). A comment on 'Some standard errors in item response theory'. Psychometrika, 49, 269-272.

Divgi, D. R. (1985). A minimum chi-square method for developing a common metric in item response theory. Applied Psychological Measurement, 9, 413-415.

Dorans, N. J. (1990). Equating methods and sampling designs. Applied Measurement in Education, 3, 3-17.

Dorans, N. J., & Kingston, N. M. (1985). The effects of violations of unidimensionality on the estimation of item and ability parameters and on item response theory equating of the GRE verbal scale. Journal of Educational Measurement, 22, 249-262.

Efron, B. (1982). The jackknife, the bootstrap and other resampling plans. Philadelphia, PA: Society for Industrial and Applied Mathematics.

Efron, B. & Tibshirani, R. J. (1993). An introduction to the bootstrap

(Monographs on Statistics and Applied Probability 57). New York: Chapman & Hall.

Eignor, D. R., Stocking, M. L., & Cook, L. L. (1990). Simulation results of effects on linear and curvilinear observed- and true-score equating procedures of matching on a fallible criterion. Applied Measurement in Education, 3, 37-52.

Feldt, L. S. & Brennan, R. L. (1989). Reliability. In R. L. Linn (Ed.), Educational Measurement (3rd ed., pp. 105-146). New York: Macmillan.

Gafni, N., & Melamed, E. (1990). Using the circular equating paradigm for comparison of linear equating models. Applied Psychological Measurement, 14, 247-256.

Haebara, T. (1980). Equating logistic ability scales by a weighted least squares method. Japanese Psychological Research, 22, 144-149.

Hambleton, R. K., & Swaminathan, H. (1985). Item Response Theory: Principles and Applications. Boston, MA: Kluwer-Nijhoff Publishing.

Hanson, B. A. (1991). A note on Levine's formula for equating unequally reliable tests using data from the common item nonequivalent groups design. Journal of Educational Statistics, 16, 93-100.

Hanson, B. A., Zeng, L., & Kolen, M. J. (1993). Standard errors of Levine linear equating. Applied Psychological Measurement, 17, 225-237.

Harris, D. J. (1991). Effects of passage and item scrambling on equating relationships. Applied Psychological Measurement, 15, 247-256.

Harris, D. J., & Crouse, J. D. (1993). A study of criteria used in equating. Applied Measurement in Education, 6, 195-240.

Hirsch, T. M. (1989). Multidimensional equating. Journal of Educational Measurement, 26, 337-349.

Huynh, H., & Ferrara, S. (1994). A comparison of equal percentile and partial credit equatings for performance-based assessments composed of free-response items. Journal of Educational Measurement, 31, 125-141.

Jarjoura, D., & Kolen, M. J. (1985). Standard errors of equipercentile equating for the common item nonequivalent populations design. Journal of Educational Statistics, 10, 143-160.

Kim, S.-H., & Cohen, A. S. (1995). A minimum chi-square method for equating tests under the graded response model. Applied Psychological Measurement, 19, 167-176.

Klein, L. W., & Jarjoura, D. (1985). The importance of content representation for common-item equating with nonrandom groups. Journal of Educational Measurement, 22, 197-206.

Kolen, M. J. (1990). Does matching in equating work? A discussion. Applied Measurement in Education, 3, 97-104.

Kolen, M. J. (1991). Smoothing methods for estimating test score distributions. Journal of Educational Measurement, 28, 257-282.

- Kolen, M. J., & Brennan, R. L. (1995). Test equating: methods and practices. New York: Springer.
- Kolen, M. J., & Harris, D. J. (1990). Comparison of item preequating and random groups equating using IRT and equipercentile methods. Journal of Educational Measurement, 27, 27-39.
- Kolen, M. J., & Whitney, D. R. (1982). Comparison of four procedures for equating the tests of general education development. Journal of Educational Measurement, 19, 279-293.
- Lautenschlager, G. J., & Park, D.-G. (1988). IRT Item detection procedures: issues of model misspecification, robustness, and parameter linking. Applied Psychological Measurement, 12, 365-376.
- Lawrence, I. M., & Dorans, N. J. (1990). Effect on equating results of matching samples on an anchor test. Applied Measurement in Education, 3, 19-36.
- Linn, R. L., Levine, M. V., Hastings, C. N., & Wardrop, J. L. (1981). An investigation of item bias in a test of reading comprehension. Applied Psychological Measurement, 5, 159-173.
- Liou, M. (1990). Effect of scale adjustment on the comparison of item and ability parameters. Applied Psychological Measurement, 14, 313-321.
- Liou, M., & Cheng, P. E. (1995). Asymptotic standard error of equipercentile

equating. Journal of Educational and Behavioral Statistics, 20, 259-286.

Little, R. J. A., & Rubin, D. B. (1994). Test equating from biased samples, with application to the Armed Services Vocational Aptitude Battery. Journal of Educational and Behavioral Statistics, 19, 309-335.

Livingston, S. A., Dorans, N. J., & Wright, N. K. (1990). What combination of sampling and equating methods works best? Applied Measurement in Education, 3, 73-95.

Lord, F. M. (1980). Applications of Item Response Theory to Practical Testing Problems (First ed.). Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Lord, F. M. (1982). Standard error of an equating by item response theory. Applied Psychological Measurement, 6, 463-472.

Lord, F. M. (1983). Small N justifies Rasch methods. In D. Weiss (Ed.), New Horizons in Testing. New York: Academic Press, 1983.

Loyd, B. H., & Hoover, H. D. (1980). Vertical equating using the Rasch model. Journal of Educational Measurement, 17, 179-193.

MacCann, R. G. (1990). Derivations of observed score equating methods that cater to populations differing in ability. Journal of Educational Statistics, 15, 146-170.

Microsoft Corporation (1997). Microsoft Visual C++, Professional Edition, version 5.0 [Computer programming language]. Seattle, WA: Author.

Mislevy, R. J., & Stocking, M. L. (1989). A Consumer's guide to LOGIST and BILOG. Applied Psychological Measurement, 13, 57-75.

Morris, C. N. (1982). On the foundations of test equating. In P. W. Holland, & D. B. Rubin (Eds), Test equating (pp. 169-191) . New York: Academic.

Oshima, T.C., Davey, T.C., & Lee, K. (1997, March). Multidimensional linking: four practical approaches. Paper presented at the annual meeting of the American Educational Research Association, Chicago, IL.

Petersen, N. S., Cook, L. L., & Stocking, M. L. (1983). IRT versus conventional equating methods: A comparative study of scale stability. Journal of Educational Statistics, 8, 137-156.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1996). Numerical Recipes in C: The art of scientific computing (2nd ed.). Cambridge: Cambridge University Press.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. Psychometrika Monograph, No. 17.

Samejima, F. (1972). A general model for free-response data. Psychometrika Monograph, No. 18.

Schmitt, A. P., Cook, L. L., Dorans, N. J., & Eignor, D. R. (1990). Sensitivity of equating results to different sampling strategies. Applied Measurement in Education, 3,

53-71.

Skaggs, G. (1990). To match or not to match samples on ability for equating: A discussion of five articles. Applied Measurement in Education, 3, 105-113.

Skaggs, G., & Lissitz, R. W. (1986). IRT test equating: Relevant issues and a review of recent research. Review of Educational Research, 56, 495-529.

Skaggs, G., & Lissitz, R. W. (1988). Effect of examinee ability on test equating invariance. Applied Psychological Measurement, 12, 69-82.

Smith, R. M., & Kramer, G. A. (1992). A comparison of two methods of test equating in the Rasch Model. Educational and Psychological Measurement, 52, 835-846.

Stocking, M. L., & Lord, F. M. (1983). Developing a common metric in Item Response Theory. Applied Psychological Measurement, 7, 201-210.

Thissen, D., & Wainer, H. (1982). Some standard errors in item response theory. Psychometrika, 47, 397-412.

Vale, C. D. (1986). Linking item parameters onto a common scale. Applied Psychological Measurement, 10, 333-344.

Wang, T., & Kolen, M. J. (1996). A quadratic curve equating method to equate the first three moments in equipercentile equating. Applied Psychological Measurement, 20, 27-43.

Wingersky, M. S., & Lord, F. M. (1984). An investigation of methods for reducing sampling error in certain IRT procedures. Applied Psychological Measurement, 8, 347-364.

Yamamoto, K., & Mazzeo, J. (1992). Item Response Theory scale linking in NAEP. Journal of Educational Statistics, 17, 155-173.

Zeng, L. (1995). The optimal degree of smoothing in equipercntile equating with postsmoothing. Applied Psychological Measurement, 19, 177-190.

Zwick, R. (1991). Effects of item order and context on estimation of NAEP reading proficiency. Educational Measurement: Issues and Practices, 10-16.