МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕВІРКИ ЯКОСТІ ТЕСТУВАННЯ НА ОСНОВІ МОДЕЛІ IRT»**

Виконала: студентка 2 курсу, групи 8.1228-з
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
Н. С. Дзундза
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри загальної математики, доцент, к.ф.-м.н. Зіновєєв І.В
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки

(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри
комп'ютерних наук
к.т.н., доцент
_____ Борю С.Ю.
(підпис)

« 29 » 05 2019 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Дзундзі Наталії Сергіївні

(прізвище, ім'я та по батькові)

1. Тема роботи Розробка програмного забезпечення для перевірки якості тестування на основі моделі IRT

керівник роботи Матвіїшина Надія Вікторівна, доцент

(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 29 » 05 2019 року № 812-с

2. Строк подання студентом роботи 17.12.2019

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Основні теоретичні засади якості тестування.
2. Основні теоретичні положення IRT теорії.
3. Програмне забезпечення для перевірки якості тестування.
4. Експериментальна перевірка ПЗ якості тестування на основі моделей IRT

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
ілюстрації до тексту роботи, графіки розрахунків, презентація

6. Консультанти розділів роботи

Розділ	**Прізвище, ініціали та посада консультанта**	**Підпис, дата**	
		завдання видав	**завдання прийняв**

7. Дата видачі завдання 29.05.2019

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	03.06.19	
2.	Збір вихідних даних.	15.06.19	
3.	Обробка методичних та теоретичних	15.07.19	
	джерел.		
4.	Розробка першого розділу.	20.07.19	
5.	Розробка другого та третього розділу.	20.08.19	
6.	Розробка четвертого розділу.	20.10.19	
7.	Оформлення та нормо контроль	10.12.19	
	кваліфікаційної роботи.		
8.	Захист кваліфікаційної роботи	08.01.20	

Студент _____ Н.С. Дзундза
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного забезпечення для перевірки якості тестування на основі моделей IRT»: 72 с., 40 рис., 57 джерел, 1 додаток.

ВАЛІДНІСТЬ, НАДІЙНІСТЬ, ОЦІНЮВАННЯ ЯКОСТІ ТЕСТУ, ТЕСТ, ТЕСТУВАННЯ, ЯКІСТЬ ТЕСТУ, C#, IRT МОДЕЛІ, ITEM RESPONSE THEORY, VISUAL STUDIO, WINDOWS FORM.

Об’єкт дослідження – програмне забезпечення для перевірки якості тестування.

Мета роботи: дослідження та розробка програмного забезпечення для перевірки якості тестування на основі моделей IRT.

Методи дослідження – аналітичний, синтез-метод, аналітико-синтетичний, практичний, порівняльний.

У кваліфікаційній роботі розглянуто основні теоретичні засади якості тестування, підходи до оцінки якості тестів, основи теоретичних положень Item Response Theory, її моделі для дихотомічних завдань (однопараметрична моделі Г. Раша, двопараметрична і трипараметрична моделі А. Бірнбаума); програмне забезпечення, необхідне для процедури статистичної обробки результатів тестування знань. Обґрунтовано створення програмного забезпечення для дослідження якості тестування на основі IRT теорії. В якості мови програмування було обрано C#. Результатом роботи є створене програмне забезпечення, що дозволяє провести перевірку якості тестування на основі моделей IRT. Програмне забезпечення, що розроблено, може бути використано вчителями загальноосвітніх шкіл та викладачами закладів вищої освіти для збалансованого та якісного формування тестових завдань.

SUMMARY

Master's Qualification Thesis «Software Development for Checking the Quality Testing Based on the Model IRT»: 72 pages, 40 figures, 57 references, 1 supplement.

VALIDITY, RELIABILITY, TEST QUALITY ASSESSMENT, TEST, TESTING, QUALITY OF THE TEST, C#, IRT MODELS, ITEM RESPONSE THEORY, VISUAL STUDIO, WINDOWS FORMS.

The object of the study is software for checking the quality of testing.

The aim of the study is research and development of software for testing the quality of testing based on IRT models.

The methods of the research are analytical, synthesis, analytical-synthetic and practical, comparative.

In the qualification paper the basic theoretical principles of test quality, approaches to test quality assessment, basics theoretical points of IRT theory and model for dichotomous problems (one-parameter G. Rasch model, two-parameter and three-parameter A. Birnbaum models); software required for the statistical processing of knowledge test results. The creation of software for testing the quality of testing based on IRT theory is justified. C # was chosen as the programming language. The result is software that allows checking the quality of testing based on IRT models. The developed software can be used by general education teachers and higher education teachers for balanced and high quality test items.

ЗМІСТ

Завдання на кваліфікаційну роботу……... 2

Реферат…….. 4

Summary……. 5

Перелік умовних скорочень……. 8

Вступ……... 9

1 Основні теоретичні засади якості тестування …… 10

1.1 Основні аспекти теорії тестування…….. 10

1.2 Підходи оцінювання якості тестів……... 13

1.3 Висновки за 1 розділом…… 18

2 Основні теоретичні положення IRT теорії ……..…… 19

2.1 Теоретичні аспекти IRT теорії ……..…….. 19

2.2 Моделі IRT теорії……. 21

2.3 Висновки за 2 розділом……... 25

3 Програмне забезпечення для перевірки якості тестування……... 26

3.1 Основні теоретичні відомості ПЗ для тестування…… 26

3.2 Assessment Systems Corporation……. 27

3.3 Winsteps…….. 32

3.4 ConQuest …… 37

3.5 LOGIST……... 39

3.6 RUMM……. 41

3.7 Висновки за 3 розділом…… 43

4 Експериментальна перевірка програмного забезпечення якості тестування на основі моделей IRT …… 44

4.1 Загальні відомості…… 44

4.2 Інструкція користувача ПЗ…….. 44

4.3 Приклад застосування програмного продукту……. 45

4.4 Висновки за 4 розділом……... 57

Висновки……..….... 58
Перелік посилань…….....…….. 59
Додаток АТекст програми……... 64

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

КАТ – комп’ютерне адаптивне тестування

ANST – American National Standards Institute

ASC – Assessment Systems Corporation

CITAS – Classical Item and Test Analysis Spreadsheet

CML – Conditional Maximum Likelihood procedure

CRT – Classical Response Theory

CTT – Classical Test Theory

IRT – Item Response Theory

JML – Joint Maximum Likelihood procedure

MML – Marginal Maximum Likelihood procedure

NCCA – National Commission for Certifying Agencies

RUMM – Rasch Unidimensional Measurement Model

Tempus –Trans-European Mobility Programme for University Studies

1PL – One-Parameter Logistik Model

2PL – Two-Parameter Logistik Model

3PL – Three-Parameter Logistik Model

ВСТУП

Розвиток сучасної теорії освітніх вимірювань неможливо уявити без відповідних математико-статистичних методів освітнього моніторингу, значна частина яких тести. Однак разом з цим потрібно пам’ятати і про якість створених тестів різними програмними продуктами.

Актуальність теми. У часи модернізації освіти контроль знань, який найчастіше здійснюється тестуванням, все більше набуває актуальності. А застосування комп’ютерного оцінювання уже сьогодні стало необхідністю. Використання спеціалізованого програмного забезпечення для перевірки якості тестування забезпечує не тільки об’єктивність оцінювання, а й прискорює зворотній зв'язок через оперативність оброблення результатів тестування.

Мета та завдання дослідження: розробка програмного забезпечення для перевірки якості педагогічних тестів за допомогою IRT теорії.

Об’єкт дослідження – програмне забезпечення для перевірки якості тестування.

Методи дослідження. У роботі використовуються такі методи, як: аналітичний, синтез, аналітико-синтетичний, практичний, порівняльний.

Практичне значення одержаних результатів. Результати роботи можуть бути використанні вчителями загальноосвітніх шкіл та викладачами закладів вищої освіти для аналізу та підвищення якості педагогічних тестів.

Структура й обсяг кваліфікаційної роботи. Робота складається з чотирьох розділів. У першому розділі наведено основні теоретичні засади якості тестування і підходи до оцінювання якості тестів. У другому розділі наведено основні положення IRT теорії. У третьому розділі розглянуто існуюче програмне забезпечення для перевірки якості тестування. У четвертому розділі наведена експериментальна перевірка розробленого програмного забезпечення для якості тестування на основі моделей IRT.

1 ОСНОВНІ ТЕОРЕТИЧНІ ЗАСАДИ ЯКОСТІ ТЕСТУВАННЯ

1.1 Основні аспекти теорії тестування

Тест як поняття в педагогічних вимірюваннях має різні визначення в залежності від погляду авторів. В роботі Л. О. Кухар і В. П. Сергієнко [1] тест визначається як інструмент, що складається з системи тестових завдань, процедури проведення тестування і створеної технології опрацювання і аналізу результатів. І. Є. Булах в роботі [2] описує тест як сукупність тестових завдань, підібраних за певними правилами для вимірювання певного кількісного показника. В роботі В. П. Беспалькова [3] тест – інструмент, який дозволяє виявити факт засвоєння отриманих знань. За М. Б. Челишковою тест визначається як сукупність завдань, для оцінки якості підготовки випробовуваного в заданій освітній галузі [4]. За визначенням В. С. Аванесова тест – система завдань зростаючої складності, що дозволяє ефективно виміряти рівень і якісно оцінити структуру підготовленості учнів [5].

У роботі будемо розглядати тест як систему завдань спеціальної форми з відповідним змістом для об’єктивного оцінювання рівня навчальних досягнень випробовуваних.

У своїй роботі Л. О. Кухар і В. П Сергієнко писали, що тестування – одночасно і метод, і результат вимірювання [1]. У роботі Г. О. Васьківської [6] тестування описується як метод і процес педагогічного вимірювання, що полягає у кількісному вимірюванні і оцінюванні рівня знань, умінь, навичок, здатностей, властивостей, якостей випробовуваних. В. І. Звонніков під тестуванням розуміє процес застосування тестів, який є частиною навчального оцінювання [7].

Розглянемо технологію створення тесту з відповідної навчальної дисципліни за А. Н. Майоровим [8]:

а) постановка цілей і завдань предмета і форм педагогічного контролю. Обсяг годин навчальної дисципліни визначає доцільність тестування і кількість тестових завдань. Цей етап є визначальним, оскільки в процесі підготовки тесту слід чітко усвідомлювати ті результати навчання, які потрібно перевірити, тобто конкретизувати передбачувані результати навчання;

б) аналіз змісту навчальної дисципліни, систематизація матеріалу, виділення функціональної і логічної структур (структурно-логічна схема дисципліни), інформаційне насичення кожного з розділів навчальної дисципліни;

в) розробка тестових завдань, експертиза змісту і форми завдань (рецензування) та коригування. Оцінку якості тестових матеріалів слід проводити за певною методикою незалежними експертами, які не брали участі в їх розробленні.

Для тестових завдань, як складових тесту, виділимо основні форми [9]:

а) завдання закритої форми (з наданими відповідями):

– на вибір однієї відповіді;
– дихотомічні (два варіанти відповіді);
– множинні (із вказаною кількістю відповідей чи не вказаною);
– на відповідність;
– на правильну послідовність;

б) завдання відкритої форми (без зазначення відповіді):

– на доповнення;
– на перелік;
– на коротку однозначну відповідь;
– на розгорнуту відповідь.

В залежності від обраного критерію тести мають свою класифікацію. Якщо розглядати тести за метою вимірювання, то вони поділяються на нормативно-орієнтовані, які дозволяють ранжувати випробовуваних за рівнем знань (для порівняння досягнень випробовуваних) і критеріально-орієнтовані,

що дозволяють з’ясувати ступінь засвоєння випробовуваних певного розділу в обраній дисципліні [10].

За цілями використання тести поділяють на [11]:

– вхідне тестування, яке проводиться на початку навчання для визначення готовності до засвоєння нових знань;

– формуюче та діагностичне тестування для визначення засвоєння матеріалу в поточному контролі;

– тематичне і підсумкове тестування для визначення результатів навчання;

– оцінка залишкових знань.

За функціональною ознакою тести поділяються на [1]:

– тести інтелекту – для аналізу рівня розумового потенціалу випробовуваного;

– тести особистості – для оцінки різних якостей особистості та її характеристики;

– тести досягнень – для визначення знань, умінь та навичок особистості;

– тести креативності – для вивчення та оцінювання творчих здібностей;

– проективні тести – для виявлення певних психічних властивостей особистості.

В залежності від методу, процесу і технології тестування може класифікуватися на: бланкове, комп’ютерне та комп’ютерне адаптивне.

Комп'ютерна система тестування знань – інформаційна система для автоматичного проведення тестування у вигляді діалогу між випробовуваним і комп'ютером, із можливістю подальшого автоматичного підрахунку результатів тестування та одержанням зведених даних [12].

КАТ – тестування за допомогою комп'ютера, під час якого надання тестових завдань залежить від відповідей екзаменованого на попередні запитання [1].

Комп'ютерні тести поділяють на такі види:

а) за структурою:

– аналоги бланкових тестів;

– власне тести;

б) за кількістю тестованих:

– тести індивідуального тестування;

– тести групового тестування (для одночасної подачі ідентичного матеріалу на комп'ютерах, об'єднаних в локальну мережу);

в) за завданням:

– діагностичні тести;

– навчальні (тести-тренажери, навчальні програми).

1.2 Підходи оцінювання якості тестування

Тестування як і будь-який метод педагогічного вимірювання ґрунтується на критеріях, які визначають якість засобу вимірювання. Якість тесту враховується в самому початку через дотримання рекомендованих вимог при розробці педагогічних тестів. В роботі [1] якість тесту характеризується як окрема категорія, яка базується на певних теоретичних і методологічних засадах, що використовуються під час аналізу результатів тестування. Головними критеріями якості тесту за В. С. Аванесовим є точність результатів вимірювання і адекватність при інтерпретації результатів [5].

Основними критеріями якості в роботі Т. М. Канівець [11] є:

– валідність (відповідність цілям контролю);

– надійність (сталість результатів тестування при багаторазовому використанні);

– репрезентативність (повнота включення вивченого матеріалу в тест);

– стандартизованість (визначення єдиної процедури проведення та підведення підсумків тестування).

Аналізу валідності, як одного з важливих критеріїв оцінки якості тестування, присвячено багато робіт. Так А. Анастазі у своїй роботі визначає валідність як поняття, яке вимірює тест і наскільки якісно це робиться [13]. А. Майоров описує валідність як характеристику того, наскільки тест відображає те, що повинен оцінювати [8]. У роботі [14] валідність вважається комплексною характеристикою, що визначається параметрами засобу, процедурами вимірювання і властивостями досліджуваної ознаки. В. Аванесов у своїй роботі [15] писав, що валідність – придатність тестових результатів для тієї мети, заради чого проводилося тестування.

Розглянемо класифікацію валідності, яка була запропонована П. Клайном у роботі [16]:

– очевидна валідність (face validity – уявлення про тест, сфери його застосування, результативності і прогностичної цінності, які виникають у випробовуваного, який не має спеціальних даних про характер використання і цілі методики);

– конкурентна валідність (concurrent validity – характеристика тесту, яка відображає його здатність відрізняти випробовуваних на основі діагностичної ознаки, яка є об’єктом дослідження);

– прогностична валідність (predictive validity – інформація про тест, яка характеризує ступінь точності і обґрунтування судження про певну латентну якість, яка діагностується за його результатами через деякий час);

– інкрементна і диференціальна валідність (incremental and differential validity відображає практичну цінність методики при проведенні відбору і здатність методики диференціювати випробовуваного за різними областями прояву якостей);

– змістовна валідність (content validity характеризує ступінь репрезентативності змісту завдань тесту по відношенню до вимірюваної області знань);

– конструктивна валідність (construct validity включає в себе всі вище перераховані види валідності).

Ще одна класифікація валідності за двома критеріями: валідність тесту (тобто валідність тестових завдань, процедури тестування і процедури оцінювання) і валідність обраного методу. Валідність методу, тобто відповідність тому, що вимірюється конкретним методом, яка в свою чергу поділяється на [1]:

– валідність змісту (ступінь впевненості, що завдання тесту охоплює весь потрібний зміст);

– валідність відповідності (відповідність результатів вимірювання та оцінювання різними методами);

– валідність прогнозу (впевненість, що отриманні оцінки добре прогнозують майбутні досягнення випробовуваного).

Для визначення валідності змісту і відповідності використовують коефіцієнт валідності (коефіцієнту Пірсона), який дорівнює коефіцієнту кореляції між результатами, одержаними різними методами за однакових умов, і показує, наскільки збігаються результати вимірів [2].

Для означення коефіцієнта Пірсона введемо такі множини параметрів: середнє вибіркове (середнє арифметичне) для двох множин $\bar{X}$, $\bar{Y}$, стандартне відхилення S_X, S_Y за множинами X і Y відповідно, коваріація S_{XY} (яка дозволяє виявити ступінь відповідності більших значень із множини X більшим значенням з множини Y – прямий зв'язок і більших значень із X меншим з Y – обернений зв'язок), які знаходяться за відповідними формулами:

$$\bar{X} = \frac{X_1 + X_2 + \cdots + X_N}{N} = \frac{\sum_{i=1}^{N} X_i}{N}, \tag{1.1}$$

$$\bar{Y} = \frac{Y_1 + Y_2 + \cdots + Y_N}{N} = \frac{\sum_{i=1}^{N} Y_i}{N}, \tag{1.2}$$

$$S_X = \sqrt{S_X^2} = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \bar{X})^2}{N-1}}, \tag{1.3}$$

$$S_Y = \sqrt{S_Y^2} = \sqrt{\frac{\sum_{i=1}^{N}(Y_i - \bar{Y})^2}{N-1}}, \quad (1.4)$$

$$S_{XY} = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{N-1}, \quad (1.5)$$

де $i = 1,2,\dots,N$.

Для розрахунку коефіцієнту Пірсона для двох наборів даних X, Y використовують формулу:

$$r_{XY} = \frac{S_{XY}}{S_X S_Y} = \frac{\sum_{i=1}^{N}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\left(\sum_{i=1}^{N}(X_i - \bar{X})^2\right)(\sum_{i=1}^{N}(Y_i - \bar{Y})^2)}}, \quad (1.6)$$

де $i = 1,2,\dots,N$.

Результат тестування можна вважати валідним при коефіцієнті кореляції більшому ніж 0,6.

Другим важливим критерієм оцінки якості тестування є надійність. В роботі Л. О. Кухар і В. П. Сергієнко надійність визначається як міра правильності і адекватності відображення тестом рівня знань випробовуваного.

За П. Клайном, надійність – характеристика методики, яка відображає точність вимірювань і стійкість результатів тесту до дії сторонніх факторів, а також внутрішня узгодженість тесту з часом. П. Клайн виділяв такі види надійності [16]:

– надійність паралельних форм (parallel-form reliability – характеристика методики за допомогою взаємодії форм тесту пропонованих одній і тій ж групі випробовуваних);

– ретестова надійність (test-retest reliability – характеристика методики при повторному дослідженні випробовуваних за допомогою одного і того ж тесту з проходженням часу);

– надійність частин тесту (split-half reliability – характеристика методики отримана шляхом аналізу стійкості результатів окремих сукупностей тестових задач або окремих завдань тесту).

В дослідженні будемо розглядати якість тесту як категорію з відповідними вимогами, що базується на закономірностях і певних теоріях, такі як: Classical Test Theory, Generalizability Theory або Item Response Theory.

В основу Classical Response Theory покладено класичний статистичний апарат для дослідження результатів вимірювань. Засновником класичної теорії тестів є Ч. Спірмен (C. Spearman), проте всебічно основи CTT вперше були викладені у роботі Г. Гулліксена (H. Gulliksen) «Theory of Mental Tests» [17] в 1950 році. Вклад в розвиток класичної теорії вніс у своїх працях [18, 19] Л. Гуттман (L. Guttman). Детальніше про вклад Л. Гуттмана описано в роботі «Louis Guttman's Contributions to Classical Test Theory» [20].

Подальший розвиток математичного апарату сприяв вдосконаленню CTT. Дослідження в цій області були викладені в роботах Ф. М. Лорд (F. M. Lord) і М. Р. Новік (M. R. Novick) [21], а також Л. Крокером (L. Crocker) і Дж. Алгино (J. Aligna) [22]. Початки дисперсійного аналізу та внутрікласової кореляції, які застосовують для оцінки величини коефіцієнта надійності тесту, були розглянуті в роботі Р. А. Фішером (R. A. Fisher) [23].

Generalizability Theory (теорія узагальнення) – це статистична теорія для оцінки надійності спостережень [24]. Ця теорія була представлена у роботі Л. Д. Кронбаха (L. J. Cronbach), Р. Нагесварі (R. Nageswari) та Д. К. Глізера (G. C. Gleser), в 1963 році [25]. Теорія узагальнення допускає і враховує нестабільність умов оцінки, які можуть вплинути на вимірювання. Перевага Generalizability Theory полягає в тому, що дослідники можуть оцінити, яка частка загальної дисперсії результатів обумовлена індивідуальними факторами, які часто варіюються в оцінці, такими як обстановка, час, предмети і оцінювачі.

В 1972 році була опублікована монографія [26], яка систематизувала та розширила попередні дослідження в цій теорії. Подальші дослідження

проводили Р. Л. Бренан (R. L. Brennan) у роботах [27-29], Д. Хуанг (J. Huang) [30, 31], Г. Солано-Флорес (G. Solano-Flores) М. Лі (M. Li) [32] та інші.

Першими поштовхами до створення Item Response Theory були дослідницькі роботи А. Біне (A. Binet) і Т. Симон (T. Simon), в яких вони графічним шляхом прагнули виявити якість завдань. Вони першими помітили закономірність, яка нагадувала криву (пізніше буде названа характеристичною).

Великий вклад в теорію IRT зробив датський математик Г. Раш (G. Rasch) у своїй роботі [33], який запропонував математичну модель, яка більш відома як однопараметрична модель Раша і ввів дві міри: «логіт рівня знань» і «логіт рівня складності завдання». Ще одним важливим внеском в теорію були роботи А. Бірнбаума (A Birnbaum), його моделі (з двома і трьома параметрами) [34].

З розвитком математичного апарату IRT тільки вдосконалювалась і була викладена в роботах Ф. М. Лорд (F. M. Lord) «Application of Item Response Theory to Practical Testing Problems» [35], Л. Крокер (L. Crocker) і Дж. Алгино (J. Aligna) [22]. Сучасні дослідження проводилися в роботах О. В. Авраменко [36] і Т. В. Лісової «Моделі та методи сучасної теорії тестів» [37] (у рамках роботи для Tempus), В. Аванесова [38-40], М. Б.Челишкової [41], В. С. Ким [42], Ю. М. Нейман та В. А. Хлебникова [43], А. П Свиридова [44].

1.3 Висновки за 1 розділом

Аналіз основних аспектів теорії тестування і підходів для оцінювання якості тестування дозволяє зробити науково обґрунтований висновок про доцільність використання тестового підходу до оцінювання рівня оволодіння теоретичними знаннями, якості практичних умінь та навичок, здатність застосовувати випробовуваними вивчений матеріал під час розв’язування практичних та теоретичних завдань.

2 ОСНОВНІ ТЕОРЕТИЧНІ ПОЛОЖЕННЯ IRT ТЕОРІЇ

2.1 Теоретичні аспекти IRT теорії

В основу Item Response Theory покладено припущення про існування деякого взаємозв’язку між спостережуваними результатами і латентними якостями тих, хто виконує тест. Латентний параметр – це параметр, що показує здібності особистості, які недоступні для прямого спостереження [37].

У IRT теорії встановлюється зв'язок між двома множинами значеннями латентних параметрів, які оцінюються за допомогою математико-статистичних моделей вимірювання. Перша множина визначає рівень підготовки випробовуваних, позначається θ_i $(i = 1,2, \dots, N)$, де i – номер того, хто проходить тест, N – кількість випробовуваних. Кожному випробовуваному відповідає лише одне значення параметра з інтервалу $(-\infty, \infty)$.

Зв’язок між отриманими оцінками і рівнем підготовки має вигляд монотонної та нелінійної S-подібної кривої, яка називається характеристичною кривою завдання [22]. Геометрична інтерпретація зображена на рис. 2.1, де по горизонтальній лінії відкладено рівень підготовки випробовуваних (параметр θ), а по вертикалі ймовірність правильної відповіді відповідно до латентної характеристики. Так при $\theta = 2$ ймовірність правильної відповіді 0,1.

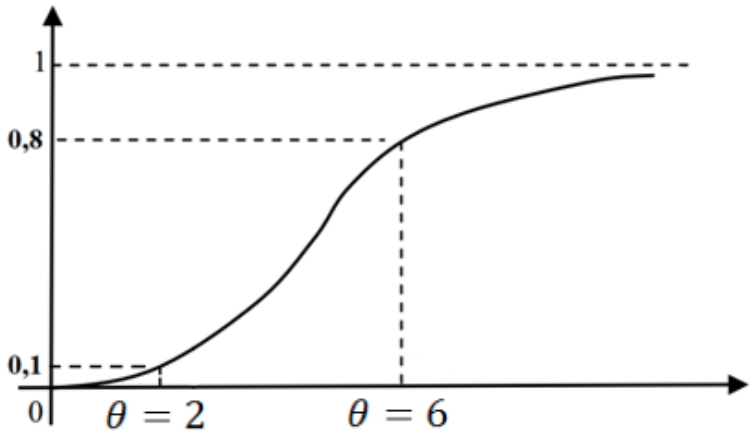

Рисунок 2.1 – Характеристична крива завдання

Друга множина характеризує складність j-го завдання і позначається β_j $(j = 1, 2, \ldots, n)$. Індекс j змінюється в межах від 1 до n, де n – кількість завдань у тесті.

Зв’язок між отриманими ймовірностями відповіді на питання в залежності від рівня складності має вигляд монотонно-спадної та нелінійної кривої, яка називається індивідуальною кривою випробовуваного (див. рис. 2.2).

Так зі складністю завдання $\beta = 4$ ймовірність виконання завдання випробовуваним дорівнює $0{,}7$, а при складності $\beta = 6$ ймовірність виконання завдання рівна $0{,}3$.

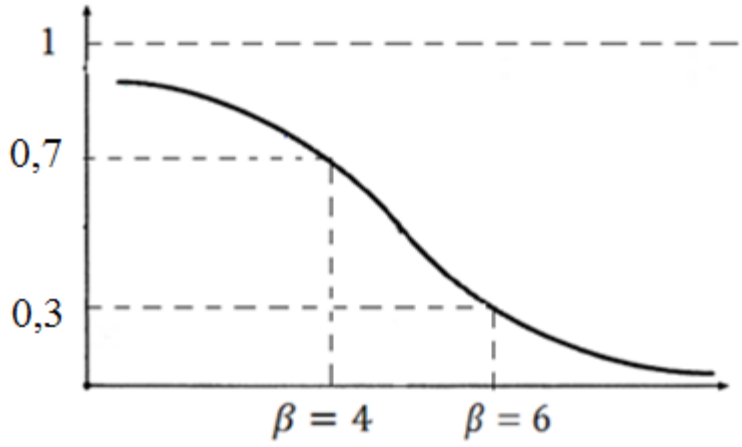

Рисунок 2.2 – Індивідуальна крива випробовуваного

Запропонована Г. Рашем математичну модель зв'язку між латентними параметрами у вигляді різниці θi – βj, при умові, що параметри оцінюються в одній шкалі дозволяє вимірювати рівень досягнень випробовуваного в спеціальних одиницях виміру – логітах [33].

При застосуванні IRT вводиться позначення умовної ймовірності P_i вірного виконання i-тим випробовуваним з рівнем підготовки θ_i, різних за складність завдань тесту, вважаючи θ_i параметром, β – незалежною змінною. Аналогічно вводиться P_j для позначення ймовірності правильного виконання j-го завдання складності β_j різними випробуваними групи, де θ – незалежна змінна, а β_j – параметр, що визначає складність j-го завдання тесту [41].

2.2 Моделі IRT теорії

Число параметрів, які входять в аналітичне задання функції, є підставою для поділу IRT на класи. Так виділяють однопараметричну модель Г. Раша та моделі А. Бірнбаума з двома та трьома параметрами [41].

Однопараметрична модель Г. Раша (1PL), яку часто називають логістичною моделлю, описана формулою [1]:

$$P_j(\theta) = \frac{e^{1,7(\theta-\beta_j)}}{1+e^{1,7(\theta-\beta_j)}}, \quad (2.1)$$

$$P_i(\beta) = \frac{e^{1,7(\theta_i-\beta)}}{1+e^{1,7(\theta_i-\beta)}}, \quad (2.2)$$

де $\theta,\ \beta$ – незалежні змінні для відповідних функції, $i = 1, \dots, N,\ j = 1, \dots, n$, N – кількість випробовуваних, n – кількість завдань у тесті.

Застосовуючи 1PL, маємо змогу встановити ймовірність виконання завдання, знаючи рівень підготовленості учасника тестування та рівень складності цього завдання. Проте в цій моделі крутизна кривих завдань вважається однаковою, через що можна зробити похибки в інтерпретації результатів, позбувшись від завдань з більш крутими характеристичними кривими, залишивши пологіші криві [7].

Для вирішення цієї проблеми використовують двопараметричну модель (2PL) А. Бірнбаума, яка включає в себе параметр диференційованої спроможності завдання тесту, що дозволяє розрізняти учасників тестування з різним рівнем навчальних досягнень.

Двопараметрична модель А.Бірнбаума для умовної ймовірності вірного виконання завдання тесту випробовуваними знаходиться за формулою [2]:

$$P_j(\theta) = \frac{e^{1,7a_j(\theta-\beta_j)}}{1+e^{1,7a_j(\theta-\beta_j)}}, \quad (2.3)$$

$$P_i(\beta) = \frac{e^{1,7a_i(\theta_i-\beta)}}{1+e^{1,7a_i(\theta_i-\beta)}}, \quad (2.4)$$

де a_j – параметр характеристики диференційованої спроможності завдання при зміні різних значень θ, a_i – параметр, що вказує на міру структурованості знань випробовуваного, що проходить тест [34].

Параметр a_j пов'язаний з крутизною кривої завдання в точці її перегину. Кориснішими є завдання у яких значення a_j більші, як на кривій a_2 (див. рис. 2.3). Крива a_1 пологіша за a_2, при інтерпретації це означає, що випробовувані з хорошим та поганим рівнем підготовки виконують дане завдання з приблизно рівною ймовірністю, а це в свою чергу не дає інформацію про індивідуальні відмінності випробовуваних [37].

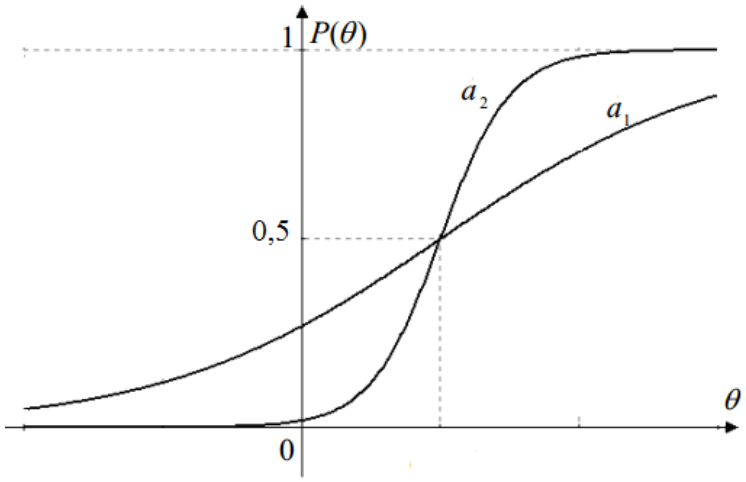

Рисунок 2.3 – Характеристичні криві завдання з різною крутизною

Параметр a_j називається параметром диференційованої спроможності j-го завдання тесту, який можна підрахувати за формулою [22]:

$$a_j = \frac{(r_{bis})_j}{\sqrt{1-(r_{bis})_j^2}}, \quad (2.5)$$

де $(r_{bis})_j$ – бісеріальний коефіцієнт кореляції, який знаходиться за формулою:

$$(r_{bis})_j = \frac{(\bar{X}_1)_j - (\bar{X}_0)_j}{S_x} \cdot \frac{(N_1)_j - (N_0)_j}{uN\sqrt{N^2 - N}}, \quad (2.6)$$

де $(\bar{X}_1)_j$ – середнє значення індивідуальних балів випробовуваних, які виконали вірно -те завдання тесту, $(\bar{X}_0)_j$ – середнє значення індивідуальних балів випробовуваних, які виконали невірно j-те завдання тесту, S_x – стандартне відхилення за множиною значень індивідуальних балів, $(N_1)_j$ – число випробовуваних, які виконали вірно j-те завдання, $(N_0)_j$ – число випробовуваних, які виконали невірно j-те завдання, N – загальна кількість випробовуваних, u – ордината нормованого нормального розподілу в точці, за якої лежить $100 \cdot \frac{N_1}{N}\%$ площі під нормальною кривою [41].

Разом з вище викладеними моделями існує трипараметрична модель (3PL) А. Бірнбаума, яка враховує ймовірність вірної відповіді на завдання в тому випадку, якщо відповідь була вгадана, а не основана на знаннях випробовуваного учасника.

Трипараметрична логістична модель А. Бірнбаума ймовірності вірної відповіді випробовуваним на j-е завдання тесту знаходиться за формулою:

$$P_j\{x_{ij} = 1 \mid \beta_j\} = c_j + (1 - c_j)\frac{e^{1,7a_j(\theta - \beta_j)}}{1 + e^{1,7a_j(\theta - \beta_j)}}, \quad (2.7)$$

де $x_{ij} = \begin{cases} 1, \text{ якщо відповідь } i \text{ випробовуваного на } j \text{ завадання вірна;} \\ 0, \text{ якщо відповідь } i \text{ випробовуваного на } j \text{ завадання невірна.} \end{cases}$

Параметр вгадування c_j визначається кількістю відповідей до закритих завдань тесту. Наприклад, для завдання з п’ятьма дистракторами за класичною теорією ймовірності $c_j = 0{,}2$, при чотирьох запропонованих дистракторами. $c_j = 0{,}25$ [41].

При геометричній інтерпретації ймовірності вірної відповіді випробовуваним за 3PL А. Бірнбаума отримаємо характеристичну криву, яка враховує диференційну здатність завдань і має горизонтальну асимптоту c_j (рис. 2.4). Це означає, що випробовувані з низьким рівнем підготовки мають ймовірність вірної відповіді не нижчу за параметр c_j.

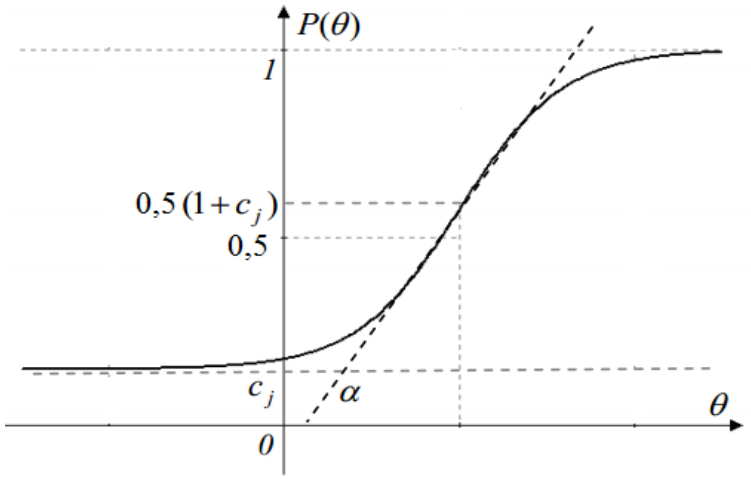

Рисунок 2.4 – Характеристична крива завдання за 3PL А. Бірнбаума

IRT дозволяє отримати числові значення рівня досягнень випробуваного в логітах за інтервальною шкалою, що дозволяє використовувати потужний апарат математичної статистики для інтерпретації отриманих результатів.

Тобто моделі Item Response Theory можна подати у вигляді схеми.

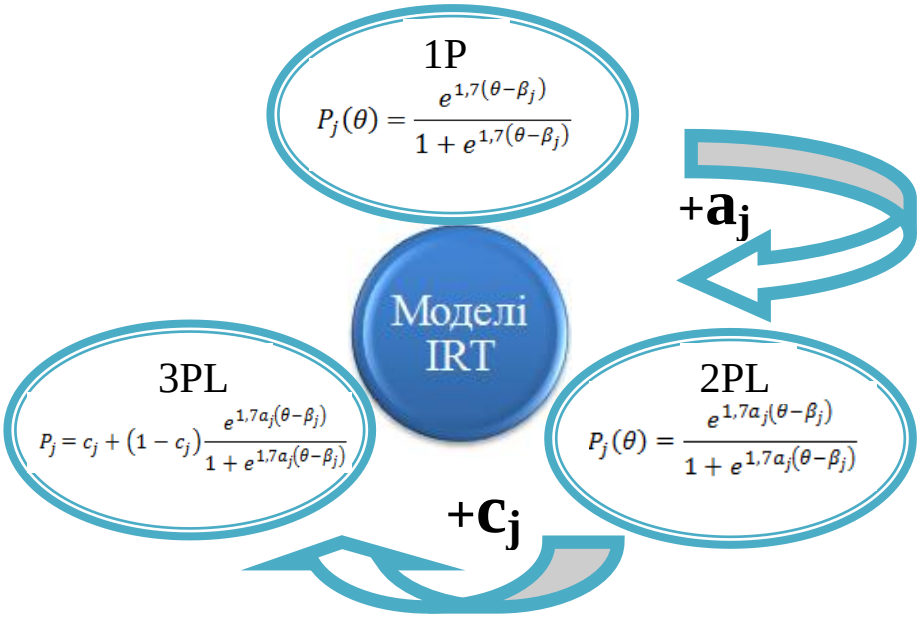

Рисунок 2.5 – Схема моделей IRT

2.3 Висновки за 2 розділом

Дослідження основних теоретичних положень Item Response Theory дає змогу дійти до висновку про доцільність використання цієї теорії для перевірки якості тестування, підвищення точності створення якісних і добре збалансованих тестів та аналізу підготовки випробовуваних.

Розглянуті моделі Item Response Theory однопараметрична модель Г. Раша, двопараметрична та трипараметрична моделі А. Бірнбаума, їх відмінності і особливості, дають змогу побудувати ансамбль характеристичних кривих за допомогою відповідних формул ймовірності для подальшої інтерпретація результатів оброблених даних.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПЕРЕВІРКИ ЯКОСТІ ТЕСТУВАННЯ

3.1 Основні теоретичні відомості ПЗ для тестування

Одним з найважливіших складових контролю знань є комп'ютерне тестування, а конкретніше – комп'ютерне адаптивне тестування. Воно дає змогу підлаштуватися під можливості того, хто проходить тест, так як наступні питання обираються залежно від відповідей на попередні питання. Тобто при правильній відповіді на питання наступне завдання береться з вищого рівня, при неправильному – із нижчого, що дає змогу визначення реального рівня знань. Однак розробка тестів для КАТ – це трудомісткий процес, головним є опрацювання банку завдань з об'єктивно визначеним рівнем складності, зв'язків між завданнями, розподілу матеріалу на базовий та додатковий і вибір програмних продуктів для аналізу якості тестів.

Програмні продукти для конструювання тестів створюються для аналізу емпіричних даних тестування з метою коригування основних характеристик тестів, забезпечення високої якості педагогічних вимірювань, вдосконалення завдань в базах даних та для побудови стандартних шкал за даними педагогічних вимірювань. Використання комп'ютерного оцінювання забезпечує не тільки автоматизацію та швидкість опрацювання результатів тестування, що в свою чергу прискорює зворотній зв'язок для виявлення прогалин в знаннях, а й надає можливість модифікувати та вдосконалювати системи тестових завдань для забезпечення об'єктивного оцінювання [45].

Існуючі на сьогодні програми переважно основані на двох теоріях: IRT та CRT. Вони дають змогу опрацьовувати великі об'ємі даних та обчислювати основні статистичні показники, такі як: характеристика тесту і тестових завдань, рівень підготовленості випробовуваних та ступінь відповідності рівня навчальних досягнень рівню складності завдань. Однак лише частина

розроблених програмних засобів для конструювання тестів безкоштовна, більшість з програм закритого типу для особистого користування. Один із лідерів в комп'ютерному тестуванні є Assessment Systems Corporation з її програмами для адаптивного тестування.

3.2 Assessment Systems Corporation

ASC була заснована в університеті штату Міннесота. Протягом 1970-х років в рамках проекту і за підтримки фінансування уряду США було розроблено комп'ютерне адаптивне тестування. Директор програми, доктор філософії Д. Вейс (D. J. Weiss) та його аспірант, доктор наук Д. Вейл (C. D. Vale), заснували Assessment Systems в 1979 році, щоб представити цю інноваційну технологію для громадськості.

Assessment Systems Corporation спеціалізується у галузі комп'ютерного тестування та психометричного програмного забезпечення на основі таких теорія як IRT та CTT. Це компанія одна з перших запропонувала громадськості професійну платформу для банківських та комп'ютеризованих тестувань у формі MICROCAT на основі DOS [46]. А вже у 2009 році ASC у співробітництві з програмною компанією 4ROI створила веб-версію своєї платформи FastTest, яка тепер застосовується у всіх основних галузях, що здійснюють оцінювання, включаючи бізнес, сертифікацію, університет, медицину та освіту К-12 (дванадцять років обов'язкової школи).

Assessment Systems Corporation допомагає при сертифікації організацій в процесі акредитації ANST – Американський національний інститут стандартів і Національною комісією із сертифікуючим агенством (NCCA).

ASC також розробила програмні забезпечення, які дозволяють публікувати звіти про професійний психометричний аналіз, використовуючи Classical Test Theory. До найбільш відомих програм Assessment Systems Corporation в основі яких лежить CTT відносяться Iteman, Lertap, CITAS.

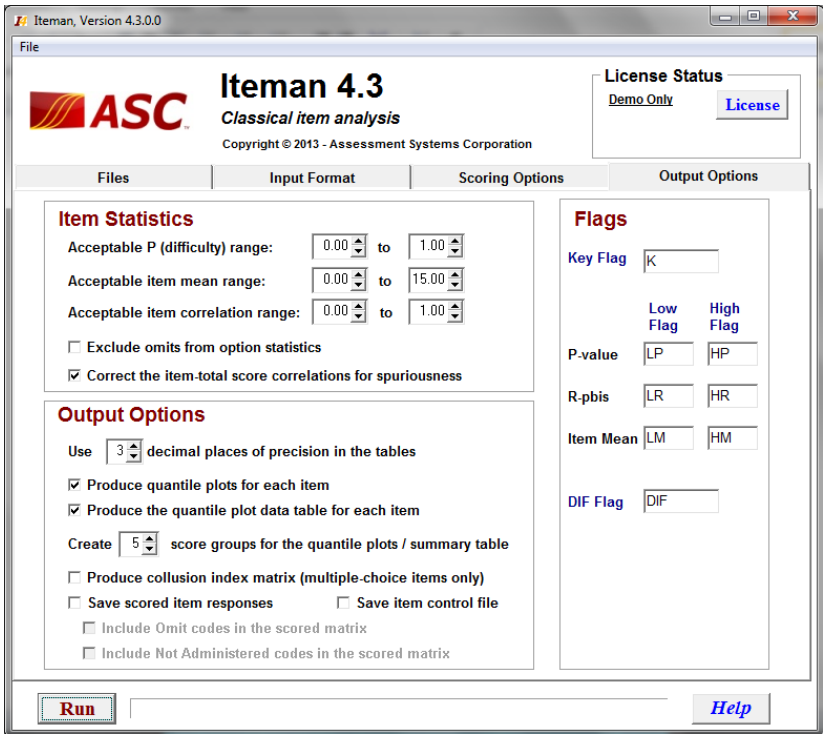

Рисунок 3.1 – Програмне забезпечення Iteman 4.3

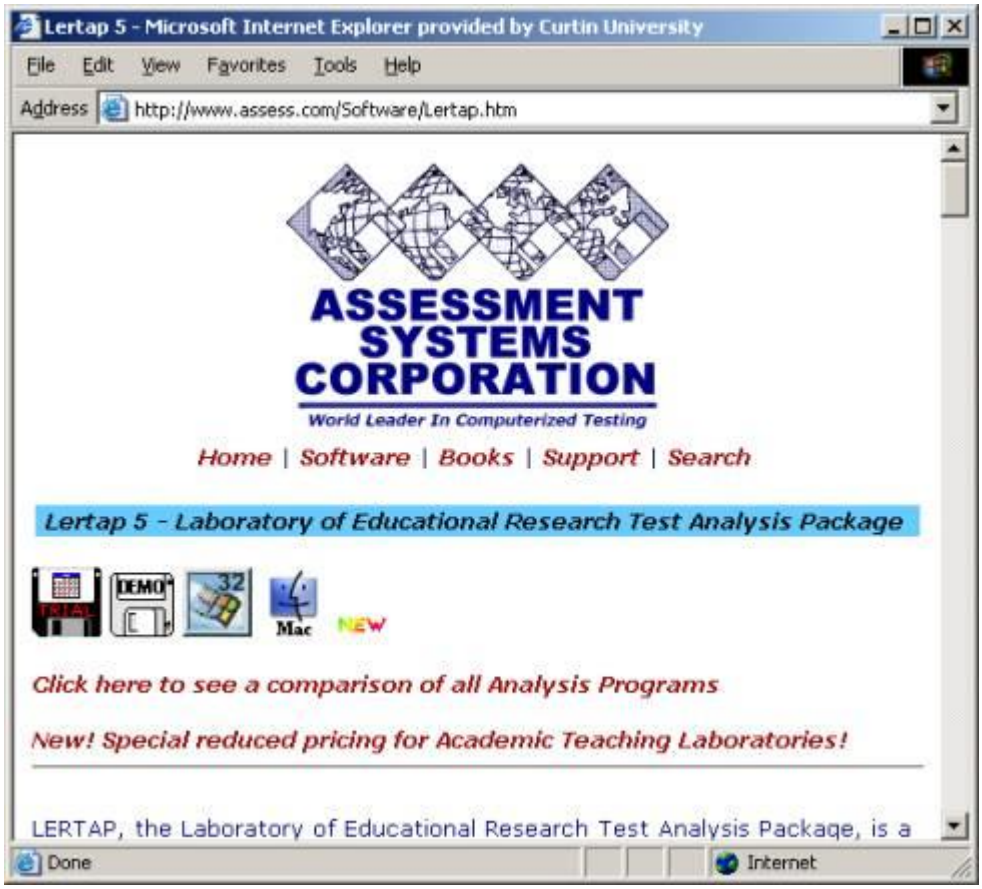

Рисунок 3.2 – Програмне забезпечення Lertap 5

В рамках цієї компанії розроблюються також програмні забезпечення, які використовують теорію IRT для обробки результатів тестувань, на них ми і звернемо увагу.

FastTest – це програмне забезпечення розроблене спеціально для використання професіоналами в галузі тестування і моделювання різних тестів, що підтримує режими бланкового і комп'ютерного пред'явлення паралельних варіантів тестів і оцінки їх якості за допомогою IRT моделей для дихотомічних даних за завданнями.

FastTest акцентує увагу на оцінку якості тестування – валідність, надійність, безпеку, масштабованість, а також забезпечує вдосконалену психометрику з повною підтримкою теорії IRT та КАТ, що дозволяє легко будувати адаптивні тести.

FastTest застосовується у всіх основних галузях, що здійснюють оцінювання, включаючи бізнес, сертифікацію, університет, медицину та освіту.

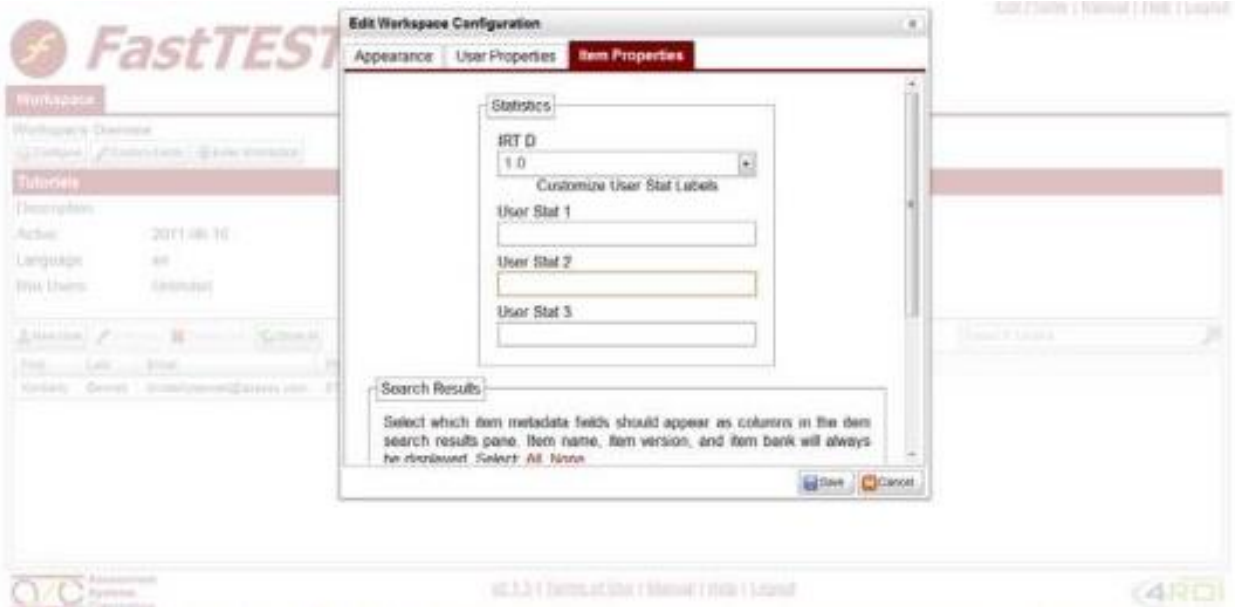

Рисунок 3.3 – Програмне забезпечення FastTest

Xcalibre – програма, створена корпорацією Assessment Systems Corporation, дозволяє отримати оцінку максимальної правдоподібності підключення на основі алгоритмів EM (Expectation-maximization algorithm) для невеликих вибірок досліджуваних або коротких тестів. Провести калібрування завдань для однопараметричної моделі Г. Раша, двопараметичної і трипараметричної моделей А. Бірнбаума в IRT теорії [47].

Xcalibre дозволяє будь-якій організації впроваджувати Item Response Theory, щоб зробити їх тести більш точними та захищеними.

Xcalibre 4 доступний у безкоштовній версії, обмеженій 50 предметами та 50 учасниками. Це дозволяє ознайомитися з програмою та почати вивчати передові методи теорії IRT.

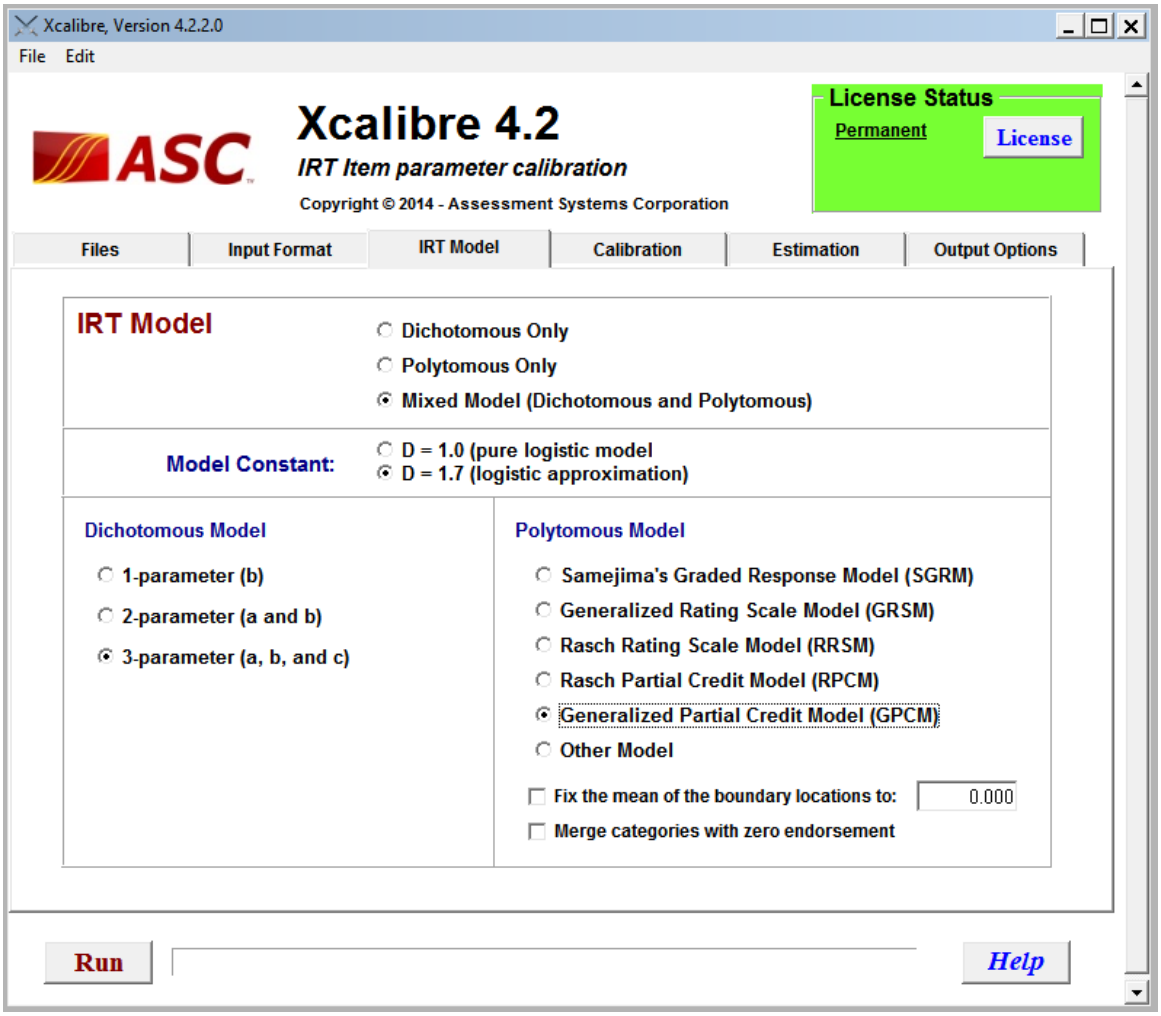

Рисунок 3.4 – Програмне забезпечення Xcalibre 4.2

Окрім Assessment Systems Corporation існують і інші виробники ПЗ для перевірки якості тестування.

BICAL – програма розроблена в 1979 році для калібрування параметрів однопараметричної моделі Г. Раша. Це широко розповсюджена програма. На сьогоднішній день існує близько 400 примірників BICAL. Однак ця програма не є безкоштовною. Оскільки BICAL не використовує алгоритм оцінки відсутніх даних то відсутні відповіді трактуються як неправильні.

Ця програма надає різноманітну інформацію про калібрування для дихотомічних даних. Наприклад, вона може проводити повторне калібрування з видаленими невідповідними випробовуваними (ті, що набрали дуже низькі та дуже високі бали) із процесу калібрування. BICAL – одна з небагатьох програм, яка надає статистику відповідності між випробовуваним та показниками в групі. Детальніше про програму написано в роботі B. D. Wright, R. J. Mead, S. R. Bell «Bical. Calibrating items with the Rasch model» [48].

BILOG – програмне забезпечення розроблене в 1984 році, дозволяє отримати оцінки параметрів тестових завдань на основі теорії IRT з використанням однопараметричної моделі Г. Раша, двопараметричної та трипараметричної моделей А. Бірнбаума для аналізу дихотомічних даних. BILOG дає змогу обробляти тести з максимум 1000 змінних на кожного випробуваного і без практичного обмеження на загальну кількість випробовуваних.

А також BILOG оцінює параметри елементів використовуючи оцінку граничної максимальної подібності (детальніше про цей метод [49]), яка використовує EM-алгоритм (Expectation-maximization algorithm) розроблений А. Демпстером (A. P. Dempster), Н. Лейрдом (N. M. Laird), та Д. Рубіномтом (D. B. Rubin) у роботі «Maximum likelihood from incomplete data via the EM algorithm» [50].

У BILOG при використанні трипараметричної моделі для оцінки параметрів використовується процедура маргінальної максимальної вірогідності MML [37].

Програмний продукт BILOG-MG є розширення програми BILOG і надає змогу отримати оцінки параметрів тестових завдань на основі для однопараметричної моделі Г. Раша, двопараметричної та трипараметричної моделей А. Бірнбаума, а також оцінити надійність і валідність тесту [51]. CITAS – дозволяє здійснити найпростішу первинну обробку даних тестування на основі класичної теорії тестів. Вхідним файлом є книга MS Excel, що містить результати тестування.

В роботі D. J. Weiss і S. V. Minden [52] описано порівняння оцінок параметрів отриманих за допомогою Xcalibre 4.1 і BILOG-MG.

Програма має обмеження на опрацювання даних (кількість іспитників та завдань не більше 100). Вихідним файлом є та ж книга з обчисленими.

3.3 Winsteps

Winsteps – програма розроблена в 1991 році під керівництвом J. M. Linacre, яка дозволяє калібрувати дихотомічні завдання за допомогою однопараметричної моделі Раша та політомічні завдання у рамках моделей [37]:

– Partial Credit Model (Masters);
– Rating Scale Model (Andrich);
– Paired Comparison Model (Bradley-Terry);
– Success Model (Glas);
– Failure Model (Linacre).

Для оцінки параметрів тут використовують процедуру максимальної вірогідності JML. Детальніше про процедуру роботи JML у Winsteps написано у роботі [53].

Дана програма набула популярності частково завдяки своїй безкоштовній академічній версії Ministep, що дає змогу аналізувати тест, складений не більше, ніж з 25 завдань, який виконувало не більше 75 випробовуваних.

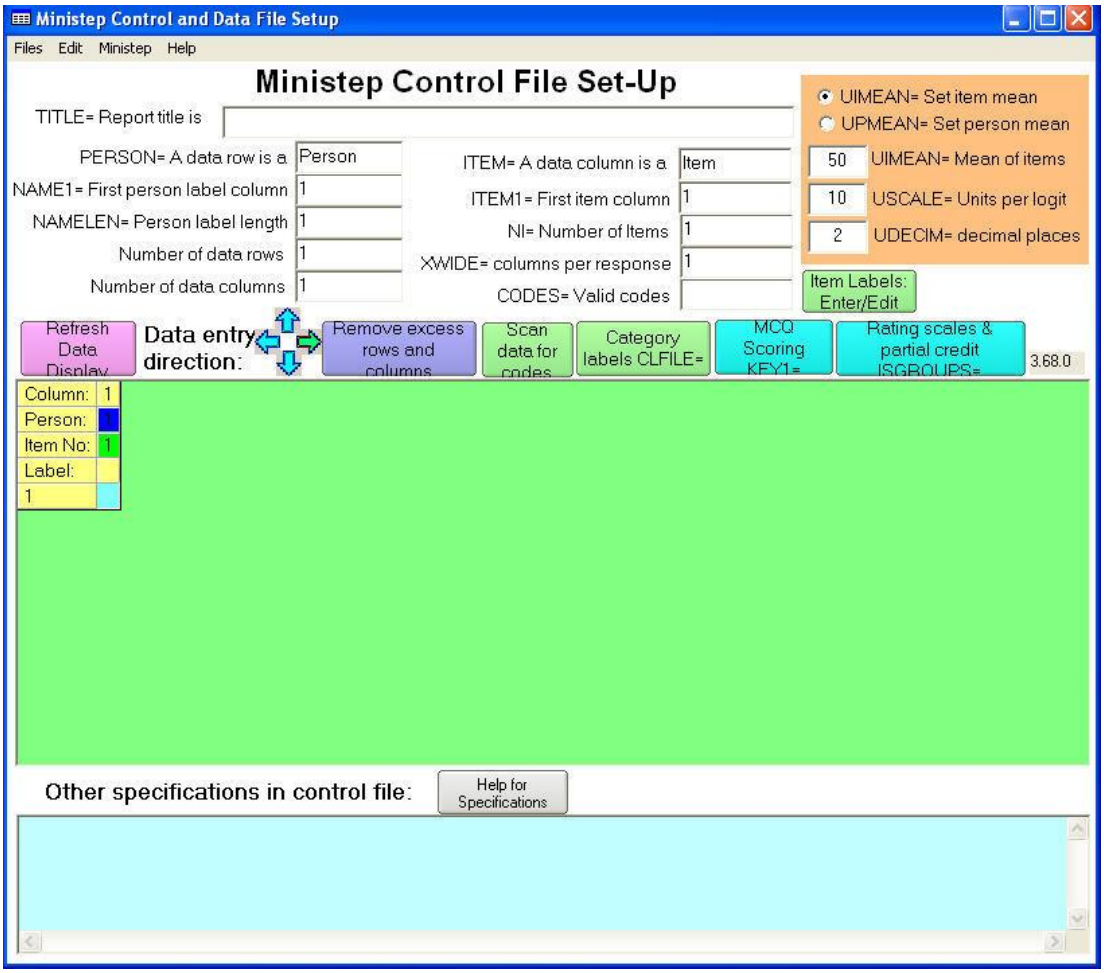

Рисунок 3.5 – Програмне забезпечення Ministep

Програмне забезпечення Ministep працює з даними в різних статистичних пакетах. Приклад застосування Ministep у форматі .xls наведено у роботі Т. В. Лісової [37].

Розглянемо детальніше роботу в Winsteps на прикладі. Для відкриття файлу з матрицею відповідей у форматі .txt натиснемо Files → Read control (and data) file і оберемо потрібний (рис. 3.6).

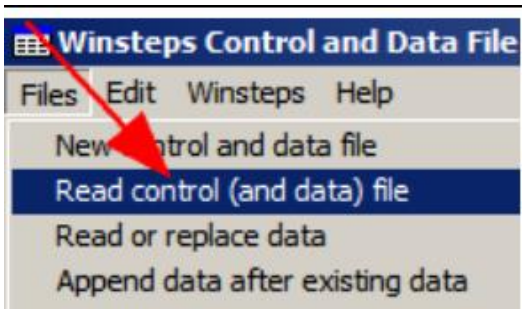

Рисунок 3.6 – Функції відкриття файлу

Після чого з’явиться екран настройки файлу управління Winsteps заповнений інформацією з документу .txt. Верхня частина екрана (обрамлена синьою рамкою) містить інструкції, що відображаються 11 керуючих змінних.

Потім вводиться кількість рядків і стовпців, щоб вхідна таблиця мала правильний розмір. У червоній рамці міститься інформацію про випробовуваного і предмет, а також відповіді. У нижній частині екрана (голубе поле) знаходяться додаткові керуючі змінні. Winsteps має близько 150 контролюючих змінних.

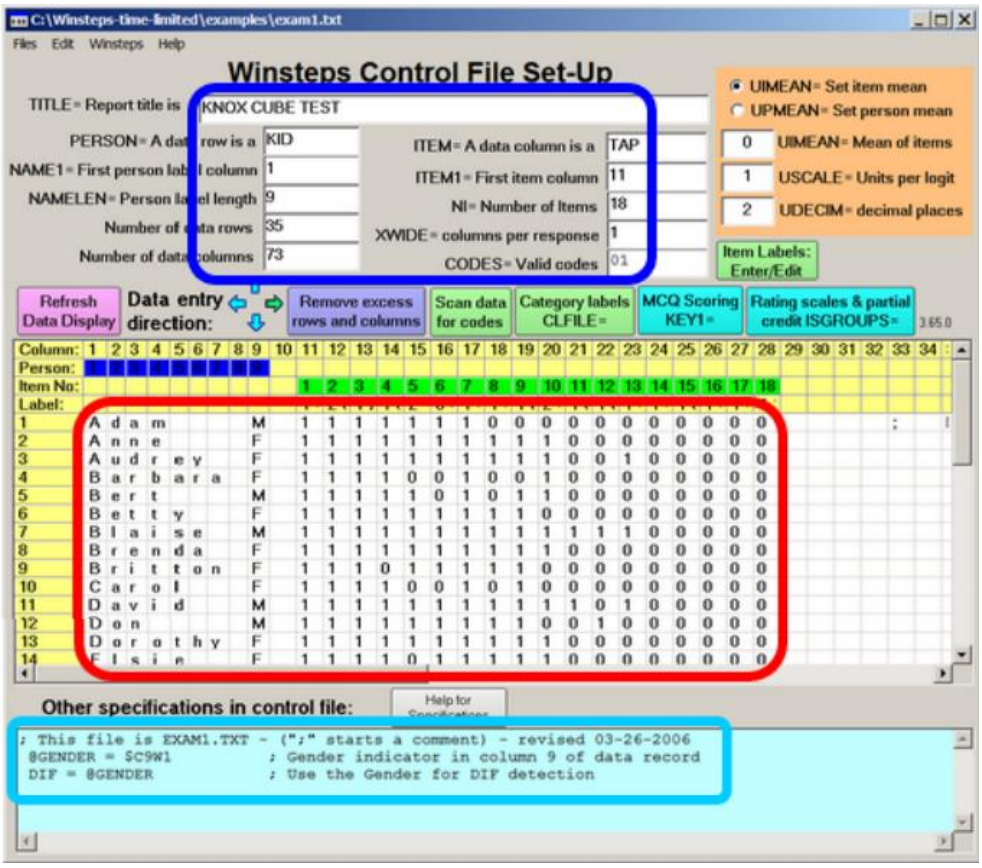

Рисунок 3.7 – Програма Winsteps з відкритими даними

Описання основних модулів текстовий інтерфейсу, призначення команд TITLE, PERSON та інших більш детальніше описано у роботі В. С. Фетісова «Комп’ютерні технології в тестуванні» [12].

Для аналізу даних у Winsteps натиснемо Enter результати наведено на рис. 3.8. Зелене поле – файл управління. В ньому відображається перші записи

даних: P – початок ярлика людини, I – початок відповідей, N – кінець відповіді. У червоному полі Winsteps починає робити свої перші оцінки.

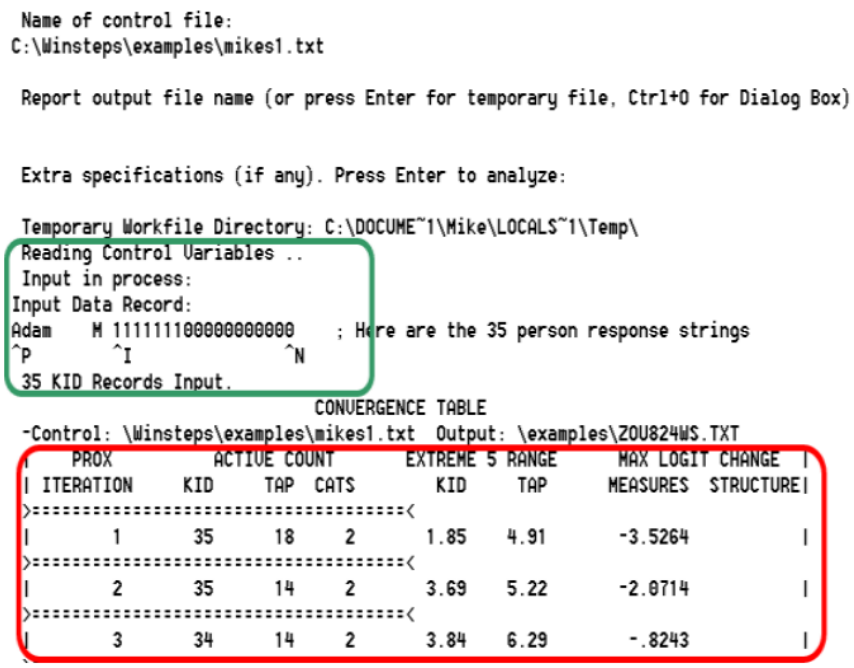

Рисунок 3.8 – Фрагмент оцінки даних у програмі Winsteps

Коли оцінка припиняється, розраховується статистика відповідності, яка зображена в зеленому полі. Статистика відповідності повідомляє, наскільки добре дані відповідають оцінкам. Узагальнені результати аналізу представлені в синьому полі (рис. 3.9).

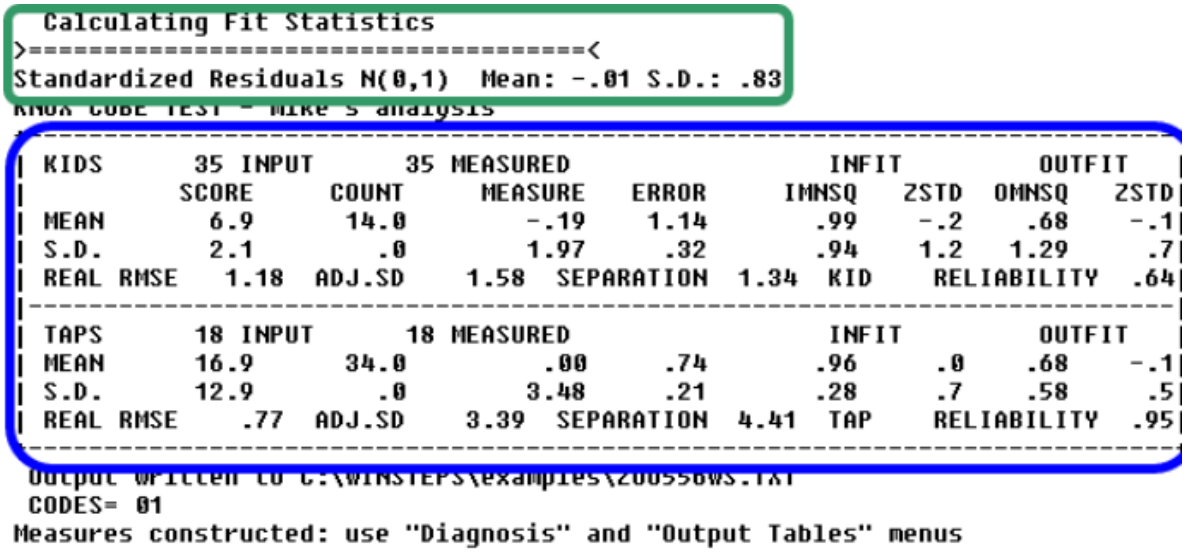

Рисунок 3.9 – Результати аналізу даних у програмі Winsteps

Для побудови графіків кривих даний з документу .txt натиснемо на меню Winsteps: Graphs (графіки) → Category Probability Curves (криві ймовірності) (див. рис. 3.10).

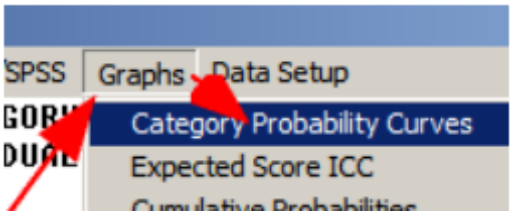

Рисунок 3.10 – Функції для побудови кривих ймовірності

Після чого відображаються криві ймовірності для заданих даних з 0 і 1 (рис. 3.11). Ось X знаходить складність в логітах, вона має нескінченний діапазон від $-\infty$ до $+\infty$. Ось Y – це ймовірність вірної відповіді. Ця вісь завжди має діапазон від 0 до 1.

На рисунку зображені червоні стрілки показують, що 1,1 логіт складності завдання відповідає 75% імовірності успіху вірної відповіді.

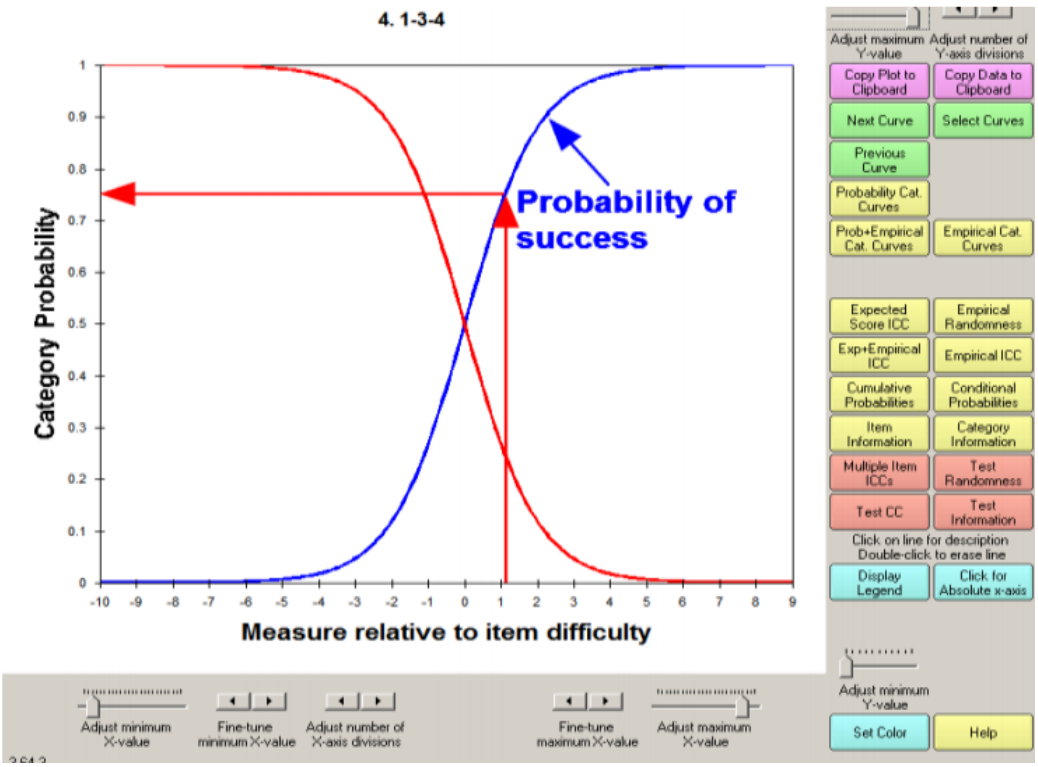

Рисунок 3.11 – Графік функції ймовірності у програмі Winsteps

3.4 ConQuest

ConQuest – комп'ютерна програма, яка надає аналіз даних, основана на всебічному та гнучкому асортименті моделей IRT (як одновимірних, так і багатовимірних) та моделей латентної регресії, що дозволяє вивчити властивості оцінок ефективності, традиційні оцінки та шкали оцінок, серед них:

– Rasch's Simple Logistic Model;
– Rating Scale Model;
– Partial Credit Model;
– Ordered Partition Model;
– Linear Logistic Test Model;
– Multifaceted Models;
– Generalized Unidimensional Model;
– Multidimensional Item Response Models;
– Latent Regression Models.

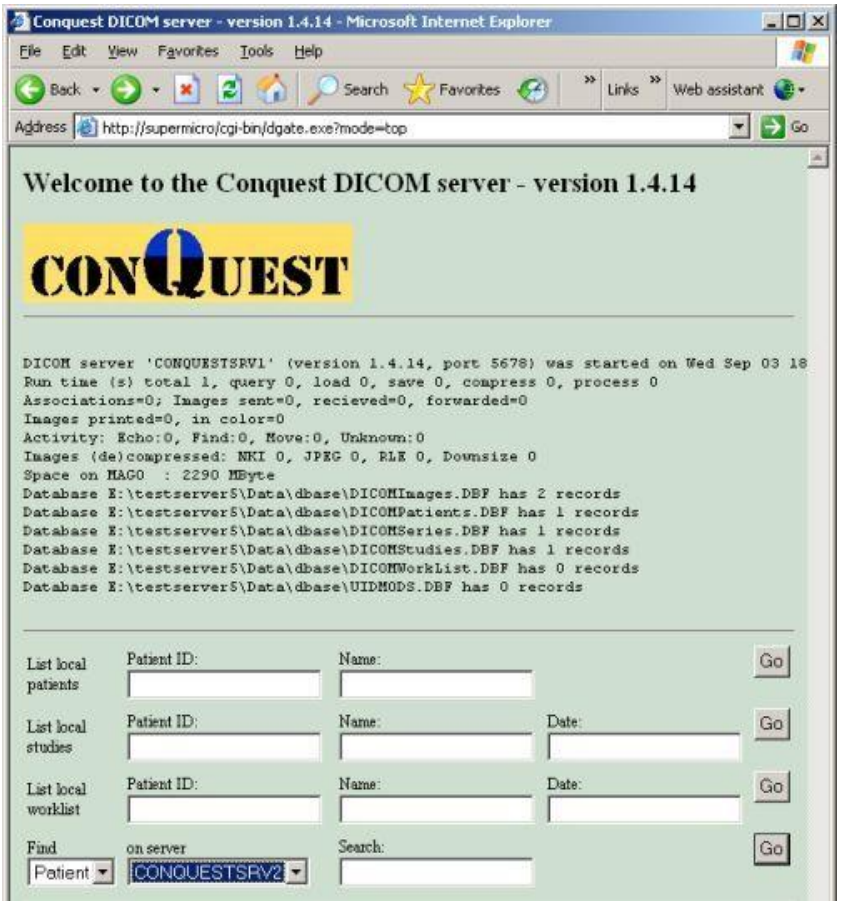

Рисунок 3.12 – Програмне забезпечення ConQuest

ACER ConQuest 4 пропонує більш широкі процедури вимірювання та дослідження аналізу процедур, що базуються на найсучасніших психометричних методах багатогранних та багатовимірних моделей IRT, моделей латентної регресії та малювання правдоподібних значень.

Авторами програми ACER ConQuest 4, яка була розроблена в компаніїї ACER у 2015 році є Р. Дж. Адамс (R. J Adams), М. Л. Ву (M. L Wu), та М. Р. Вілсон (M. R Wilson). ACER ConQuest 4 має широкий спектр Інтернет-ресурсів та доступ до швидкої підтримки електронної пошти.

ACER ConQuest 4 доступний як з графічним інтерфейсом користувача (GUI), так і з простим командним рядком або консоллю. Програмне забезпечення доступне у версіях Windows та Mac. Windows містить GUI і консольну версію для 32-бітної та 64-бітної операційних систем Windows.

Mac OS містить лише консольну версію для 64-бітної операційної системи Mac (GUI не доступний).

Особливості ACER ConQuest 4:

– оцінка моделі Бредлі-Террі-Люсі (BTL) для парних порівнянь;

– вибір граничної максимальної вірогідності або спільної оцінки максимальної вірогідності;

– встановлення багатовимірної та багатогранної форми номінальної моделі Бока, включаючи її особливі випадки, двопараметричну логістичну модель (2PL) та узагальненої часткової кредитної моделі (GPCM);

– безпосереднє зчитування системних файлів SPSS;

– виведення результатів у файли SPSS або EXCEL;

– широкий набір графічних результатів, включаючи Wright maps та карти Wright predicted;

– моделювання латентної змінної;

– додавання матричних змінних та функцій маніпулювання матрицею: обчислити, якщо і під час команд для управління матрицями;

– оцінки Mantel-Haenszel DIF.

В ConQuest версія 4.2 додано параметр «uniquepid = yes» у команді SET, різко скорочуючи час обробки наборів даних з унікальними PID (тобто кожен запис відповідає лише одному випадку PID).

В версії ConQuest 4.5 варіант десяткових знаків може бути додано до команди print для управління кількістю десяткових знаків для цифр на дисплеї. Тепер аргументи print та scatter тепер можуть бути матричними змінними.

Додано набір із 6 логічних операторів для використання в обчисленнях «= =», «! =», «<=», «>=», «>» та «<». Ці оператори застосовуються по елементу до пари матриць і повертають матриці «1» і «0» з 1, якщо порівняння елементів є істинним, а 0 – якщо помилковим.

Нова функція маніпулювання матрицею iif, яка бере три аргументи, (наприклад, iif (x, y, z)). Усі три аргументи повинні бути матрицями однакових розмірів. Результатом є матриця, де елемент бере своє значення з другого аргументу, якщо відповідний елемент «1» у першому аргументі, в іншому випадку він бере значення з другого аргументу.

3.5 LOGIST

LOGIST – програма розроблена в 1976 році, для оцінки можливостей випробовуваного і характеристик кривої елементів за методом максимальної вірогідності на основі трипараметричної моделі А. Бірнбаума.

Вона дає змогу одночасно оцінювати параметри складності завдань та параметри підготовленості іспитників за сумісною процедурою максимальної вірогідності JML. Недоліком даної процедури є вимога великої вибірки іспитників (не менше 1000) для забезпечення більш точних оцінок [34].

Розглянемо застосування програми для обробки результатів тестування. Якщо потрібно знайти оцінку параметрів тільки з даних відповіді, LOGIST оцінює параметри для кожного предмета і для кожного випробуваного, як максимум JML і оцінює процедуру в чотири кроки (див. рис. 3.13).

Parameter Status in LOGIST Estimation Steps

Step	Parameter				
	θ	a	b	c	COMC
1	estimated	fixed	estimated	fixed	not used
2	fixed	estimated	estimated	estimated	estimated
3	estimated	fixed	estimated	fixed	fixed
4	fixed	estimated	estimated	estimated	fixed

Рисунок 3.13 – Таблиця стану параметрів

Детальніше щодо процедури можна знайти в керівництві для користувача LOGIST. В програмі оцінка відбувається в діапазоні від -3 до +3, так що значення θ в цьому діапазоні мають середнє значення від 0 до стандартне відхилення 1.

Підраховуючи відповідні параметри програма дає результати у вигляді рисунків (див. рис.3.14) на основі яких і робиться аналіз тестових завдань і тесту в цілому.

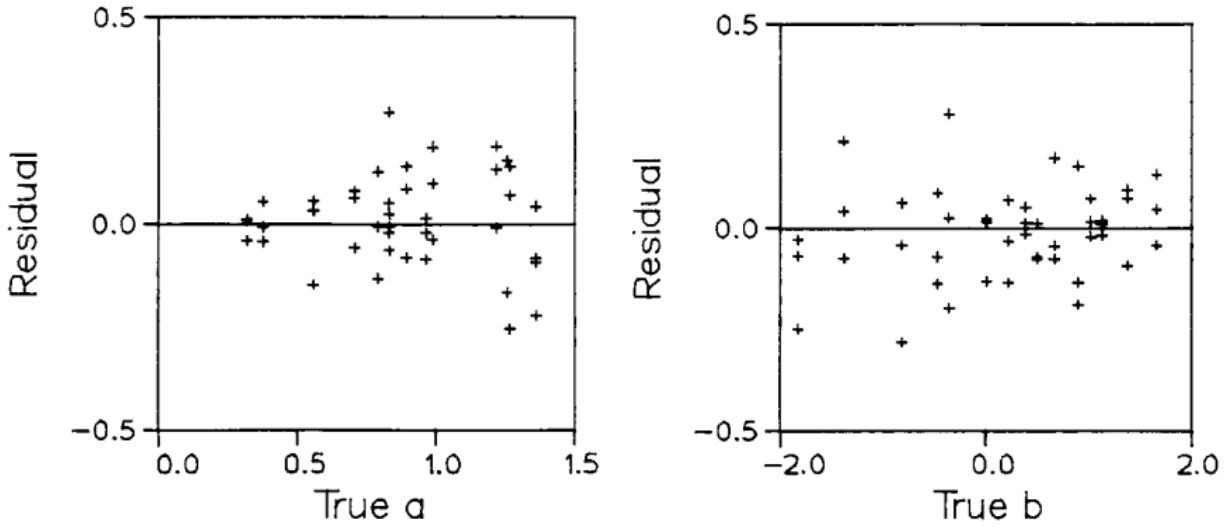

Рисунок 3.14 – Оцінки параметрів елементів для 45 завдань тесту

Якщо робити порівняльний аналіз часу роботи двох програм LOGIST і BILOG, які виконували обробку параметрів на основі трипараметричної моделі А. Бірнбаума, то можна помітити переваги LOGIST над BILOG (при настройках за замовчуванням). Перевага становить близько 2 : 1 для короткого тесту і 1,5: 1 для довшого тесту (див. рис. 3.15) [54].

Execution Times of LOGIST and BILOG on Two Simulated Datasets

Test and Program	CPU Seconds
15-Item Test	
LOGIST	19.45
BILOG (assuming data contain no omitted or not-presented items)	20.14
BILOG (assuming data may contain omitted and not-presented items)	39.32
45-Item Test	
LOGIST	37.29
BILOG (assuming data contain no omitted or not-presented items)	34.26
BILOG (assuming data may contain omitted and not-presented items)	55.58

Рисунок 3.15 – Порівняльна таблиця часу виконання двох змодельованих наборів даних програмами LOGIST і BILOG

В статті [54] більш детальніше описується порівняння теоретичних підходів двох програм LOGIST, BILOG і приклади, які ілюструють їх відмінності і подібності.

3.6 RUMM

Rasch Unidimensional Measurement Model – програмне забезпечення розроблене під керівництвом Д. Ендрича (D. Andrich) в 1990 році для оцінки якості тестових завдань, яке дозволяє оптимізувати зміст тесту і перетворити його в інструмент для вимірювання рівня знань випробовуваних [55].

У статті В. С. Кіма [56] був проведений аналіз результатів тестування за допомогою програми RUMM 2010. Подальше дослідження стосувалися роботи в RUMM 2020 (випущена в 2003 р.), що дає можливість аналізувати параметри тестових завдань відповідно до ймовірнісної моделі Г. Раша за спрощеними алгоритмами обчислень параметрів. Алгоритм опрацювання результатів тестування програмою RUMM 2020 можна знайти в роботі Г. Смирнової [57].

Вдосконалене програмне забезпечення RUMM, тобто RUMM 2030, для проведення аналізу елементів за моделлю Г. Раша було розроблено 1 січня 2010 року, а настройка ліцензії було реалізована 1 березня 2012 року.

RUMM 2030 доступний в двох версіях: стандартної і професійної. Як і при застосуванні RUMM 2020, стандартна і професійна версії доступні в якості одноразової покупки (тобто непотрібно продовжувати ліцензію в майбутньому).

Основні додаткові функції стандартної версії включають в себе:

– оцінка розмірності;

– додаткові деталі в тестовому рівнянні;

– створення наборів даних тільки з повними записами даних (у разі випадкових пропущених даних);

– розширення відображення коефіцієнтів (для повних записів даних).

Професійна версія RUMM 2030 є значним розширенням стандартної версії і надає повний спектр стратегій, процедур і графічних дисплеїв для проведення поглибленого аналізу.

Основні додаткові функції, включені в професійну версію:

– надання стандартних помилок;

– фасетний аналіз до 3-х сторонньої структури відповіді елемента;

– стратегія вивчення залежності відповіді між предметами.

– умовне випробування на придатність для пари політомічних предметів або пари випробувань;

– спеціальний аналіз відповідей, наприклад, для перевірки значущості здогадок;

– додавання кривих характеристик випробовуваного і стандартних залишкових графіків для відповідей окремих осіб по окремих елементах.

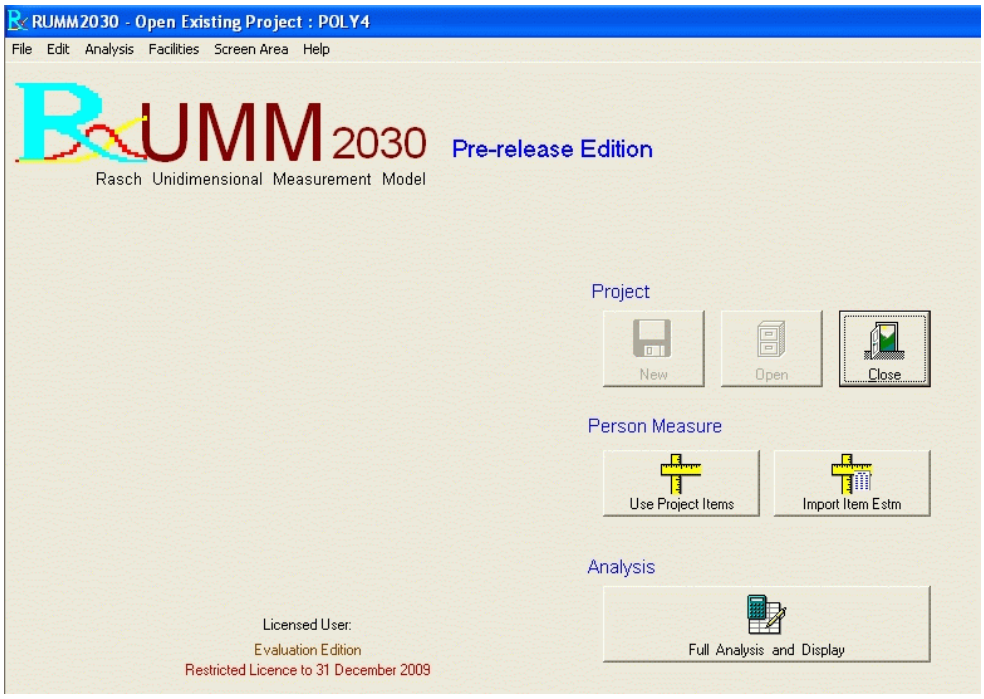

Рисунок 3.16 – Програмне забезпечення RUMM 2030

3.7 Висновки за 3 розділом

Розглянуті основні програмні засоби необхідні для процедури статистичної обробки результатів тестування знань: FastTest, Xcalibre, Winsteps, ConQuest, LOGIST, RUMM та інші. Кожна з яких має свої особливості і обмеження (іншомовні, платні та обмеження в кількості випробовуваних або тестових задань) не дають змогу в повній мірі реалізувати можливості аналізу тестування на основі моделей Item Response Theory для створення збалансованого та якісного тесту.

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ЯКОСТІ ТЕСТУВАННЯ НА ОСНОВІ МОДЕЛЕЙ IRT

4.1 Загальні відомості

Назва програми – «Перевірка якості тестування на основі моделі Item Response Theory».

Програма розроблена за допомогою Visual Studio 2017, на мові C#. Мова C# як засіб програмування була обрана через ряд своїх переваг: очевидна організованість, логічність і зручність її конструкцій, розвиненість засобів діагностики і редагування коду, які роблять процес програмування більш ефективним.

Розроблена програма призначена для перевірки якості тестування на основі моделей IRT: однопараметричної моделі Г. Раша, двопараметричної і трипараметричної моделей А. Бірнбаума, для дихотомічних завдань.

4.2 Інструкція користувача ПЗ

Програмне забезпечення може використовуватися при аналізі будь-яких результатів тестування за наявності комп’ютеру з операційною системою Windows 7 та вище.

Дані повинні відповідати строгим критеріям:

– занесені до Excel;

– лист Excel з потрібними даними, має бути 1 листом в документі;

– правильні відповіді позначаються «1», а неправильні «0»;

– аналіз не виконується за умови всіх правильних, або всіх неправильних відповідей випробовуваного чи по завданню тесту.

4.3 Приклад застосування програмного продукту

Розглянемо технологію застосування програмного забезпечення на конкретному прикладі тесту з дисципліни «Математичний аналіз» з 12 завдань в групі з 15 студентів, з метою оцінки латентних параметрів випробовуваних і параметрів завдань тесту та побудови ансамблю характеристичних кривих однопараметричної моделі Раша, двопараметричної та трипараметричної моделей Бірнбаума для аналізу якості педагогічного тесту.

Розглянемо алгоритм розрахунків латентних параметрів і побудови графіків ансамблю характеристичних кривих:

а) результати тестування оформлюються у вигляді матриці з елементами x_{ij} ($x_{ij} = 1$, якщо відповідь i-го студента на j-е завдання вірна, $x_{ij} = 0$ – якщо відповідь невірна) у документі Excel. Отримання даних з листа під назвою «Items», для реалізації однопараметричної моделі Раш, відбувається за допомогою коду описаного на рисунку 4.1;

```
private void Open()
{

    try
    {
        OpenFileDialog openFileDialog1 = new OpenFileDialog();

        openFileDialog1.InitialDirectory = "d:\\";
        openFileDialog1.Filter = "Database files (*.xls;*xlsx)|*.xls;*xlsx";
        openFileDialog1.FilterIndex = 0;
        openFileDialog1.RestoreDirectory = true;

        dataGridView1.DataSource = null;

        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            string selectedFileName = openFileDialog1.FileName;
            MessageBox.Show("Відкриваємо файл " + selectedFileName.ToString());

            string name = "Items";

            string constr = @"provider = Microsoft.ACE.OLEDB.12.0;
                    data source =" + selectedFileName +
                    ";Extended Properties = 'Excel 12.0;HDR=NO'";
            OleDbConnection con = new OleDbConnection(constr);
            OleDbCommand oconn = new OleDbCommand("Select * From [" + name + "$]", con);

            con.Open();
            OleDbDataAdapter sda = new OleDbDataAdapter(oconn);
            DataTable data = new DataTable();
            sda.Fill(data);
            dataGridView1.DataSource = data;
            con.Close();
            dataGridView1.AutoResizeColumns();
            dataGridView1.RowHeadersWidth = 70;
            for (int i = 0; i < dataGridView1.Columns.Count; i++)
                dataGridView1.Columns[i].HeaderText = "Ж" + Convert.ToString(i + 1);
            for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
                dataGridView1.Rows[i].HeaderCell.Value = "С " + Convert.ToString(i + 1);
        }
    }
```

Рисунок 4.1 – Код функції отримання даних

б) підрахунок долей вірних відповідей i-го студента і долей невірних відповідей за формулами (4.1) і (4.2).

$$p_i = \frac{X_i}{n}, \tag{4.1}$$

$$q_i = 1 - p_i, \tag{4.2}$$

де $i = 1,2, \dots, N$ (N – кількість студентів n – число завдань в тесті);

в) проводиться початкова оцінка значень параметра, що характеризує рівень підготовки студентів в логітах за формулою:

$$\theta_i^0 = ln\frac{p_i}{g_i}, \tag{4.3}$$

де p_i, q_i – долі відповідних вірних і невірних відповідей i-го студента на завдання тесту.

Реалізація підрахунку початкових оцінок значень параметра θ (рівень підготовки студента) показано на рисунку 4.2.

```
for (int i = 0; i < dataGridView1.RowCount - 1; i++)
{
    listBox1.Items.Add(SummaRows(i));
}

for (int i = 0; i < listBox1.Items.Count; i++)
{
    double pi = Convert.ToDouble(listBox1.Items[i]) / dataGridView1.ColumnCount;
    listBox2.Items.Add(pi);
}

for (int i = 0; i < listBox2.Items.Count; i++)
{
    double qi = 1 - Convert.ToDouble(listBox2.Items[i]);
    listBox3.Items.Add(qi);
}

double sumoi = 0;
for (int i = 0; i < listBox3.Items.Count; i++)
{
    double oi = Math.Log(Convert.ToDouble(listBox2.Items[i]) / Convert.ToDouble(listBox3.Items[i]));
    listBox4.Items.Add(oi);
    sumoi += oi;
}
```

Рисунок 4.2 – Код підрахунку початкових значень параметра θ

г) підраховується середнє значення логітів рівня підготовки $\bar{\theta}$ для множини θ_i^0 за формулою (4.4) і дисперсію для множини θ_i^0 за формулою (4.5).

$$\bar{\theta} = \frac{\sum_{i=1}^{N} \theta_i^0}{N}, \tag{4.4}$$

$$V = \frac{\sum_{i=1}^{N} (\theta_i^0)^2 - N(\bar{\theta})^2}{N-1}. \tag{4.5}$$

Реалізація підрахунку показано на рисунку 4.3.

```
double sumoi2 = 0;
for (int i = 0; i < listBox4.Items.Count; i++)
{
    double oi2 = Math.Pow(Convert.ToDouble(listBox4.Items[i]), 2);
    listBox5.Items.Add(oi2);
    sumoi2 += oi2;
}
textBox3.Text = Convert.ToString(sumoi2);

double oo = sumoi / listBox4.Items.Count;
textBox4.Text = Convert.ToString(oo);

double vdispersia = (sumoi2 - (listBox5.Items.Count * Math.Pow(oo, 2))) / (listBox5.Items.Count - 1);
textBox5.Text = Convert.ToString(vdispersia);
```

Рисунок 4.3 – Код розрахунку дисперсії і середнє значення логітів $\bar{\theta}$

д) підраховуються долі вірних p_j і невірних q_j відповідей на кожне завдання тесту за формулами (4.6) і (4.7).

$$p_j = \frac{Y_j}{N}, \tag{4.6}$$

$$q_j = 1 - p_j, \tag{4.7}$$

де $j = 1, 2, \dots, n$, (n – кількість завдань у тесті, N – кількість студентів).

е) проводиться початкова оцінка значень параметра складності завдань тесту за формулою:

$$\beta_j^0 = ln\frac{q_j}{p_j}, \tag{4.8}$$

де p_j, q_j – долі вірних і невірних відповідей на j-е завдання тесту.

Реалізація підрахунку початкових оцінок значень параметра β (складність завдань тесту) показано на рисунку 4.4.

```
for (int i = 0; i < dataGridView1.ColumnCount; i++)
    listBox6.Items.Add(SummaColums(i));
for (int i = 0; i < listBox6.Items.Count; i++)
{
    double pj = Convert.ToDouble(listBox6.Items[i]) / (dataGridView1.RowCount - 1);
    listBox7.Items.Add(pj);
}

for (int i = 0; i < listBox7.Items.Count; i++)
{
    double qi = 1 - Convert.ToDouble(listBox7.Items[i]);
    listBox8.Items.Add(qi);
}

double sumoj = 0;
for (int i = 0; i < listBox8.Items.Count; i++)
{
   double oj = Math.Log(Convert.ToDouble(listBox8.Items[i]) / Convert.ToDouble(listBox7.Items[i]));
   listBox9.Items.Add(oj);
   sumoj += oj;
}
```

Рисунок 4.4 – Код підрахунку початкових значень параметра β

ж) підраховуються середнє значення $\bar{\beta}$ для множини початкових значень складності β_j^0 $(j = 1, 2, \dots, n)$ за формулою (4.9) і дисперсію (4.10).

$$\bar{\beta} = \frac{\sum_{j=1}^{n} \beta_j^0}{n}, \tag{4.9}$$

$$U = \frac{\sum_{i=1}^{n} (\beta_j^0)^2 - n(\bar{\beta})^2}{n-1}. \tag{4.10}$$

Реалізація підрахунку показано на рисунку 4.5.

```
double sumoj2 = 0;
for (int i = 0; i < listBox9.Items.Count; i++)
{
    double oj2 = Math.Pow(Convert.ToDouble(listBox9.Items[i]), 2);
    listBox10.Items.Add(oj2);
    sumoj2 += oj2;
}
textBox6.Text = Convert.ToString(sumoj2);

double bo = sumoj / listBox8.Items.Count;
textBox7.Text = Convert.ToString(bo);

double udispersia = (sumoj2 - (listBox10.Items.Count * Math.Pow(bo, 2))) / (listBox10.Items.Count - 1);
textBox8.Text = Convert.ToString(udispersia);
```

Рисунок 4.5 – Код розрахунку дисперсії і середнє значення логітів $\bar{\beta}$

з) підраховуються поправочні коефіцієнти за формулами (4.11), (4.12) і оцінки параметрів θ, β в єдиній інтервальній шкалі за формулою (4.13).

$$X = \sqrt{\frac{1 + U/2.89}{1 - UV/8.35}}, \tag{4.11}$$

$$Y = \sqrt{\frac{1 + V/2.89}{1 - UV/8.35}}, \tag{4.12}$$

$$\theta_i = \bar{\beta} + X\theta_i^0,\ \beta_j = \bar{\theta} + Y\beta_j^0, \tag{4.13}$$

де всі позначення залишаються колишніми, а параметри θ і β мають оцінки $\theta_i (i = 1,2, \dots, N)$ і $\beta_j (j = 1, 2, \dots, n)$ в стандартній інтервальній шкалі.

Реалізація підрахунку формул наведених вище показано на рисунку 4.6.

```
double xkut = Math.Sqrt((1 + (udispersia / 2.89)) / (1 - ((vdispersia * udispersia) / 8.35)));
textBox9.Text = Convert.ToString(xkut);

double ykut = Math.Sqrt((1 + (vdispersia / 2.89)) / (1 - ((vdispersia * udispersia) / 8.35)));
textBox10.Text = Convert.ToString(ykut);

double paramx = 0;
for (int i = 0; i < listBox4.Items.Count; i++)
{
    paramx = (xkut * Convert.ToDouble(listBox4.Items[i])) + bo;
    listBox11.Items.Add(paramx);
}

double paramy;
for (int i = 0; i < listBox9.Items.Count; i++)
{
    paramy = (ykut * Convert.ToDouble(listBox9.Items[i])) + oo;
    listBox12.Items.Add(paramy);
}
```

Рисунок 4.6 – Код розрахунку поправочних коефіцієнтів і оцінок параметрів рівня підготовки θ і складності завдань β в єдиній інтервальній шкалі

и) знаходяться похибки вимірювання $S(\theta_i)$ параметру рівня підготовки для кожного значення θ_i $(i = 1,2, \dots, N)$ і похибки вимірювання складності j-го завдання $S(\beta_j)$ для кожного β_j за формулами (4.15) і (4.16).

$$S(\theta_i) = \frac{X}{\sqrt{np_iq_i}}, \tag{4.15}$$

$$S(\beta_j) = \frac{Y}{\sqrt{Np_jq_j}}. \tag{4.16}$$

Реалізація підрахунку формул показано на рисунку 4.7.

```
double errorx;
for (int i = 0; i < listBox2.Items.Count; i++)
{
    errorx = xkut / (Math.Sqrt(dataGridView1.ColumnCount * Convert.ToDouble(listBox2.Items[i]) * Convert.ToDouble(listBox3.Items[i])));
    listBox13.Items.Add(errorx);
}

double errory;
for (int i = 0; i < listBox7.Items.Count; i++)
{
    errory = ykut / (Math.Sqrt((dataGridView1.RowCount - 1) * Convert.ToDouble(listBox7.Items[i]) * Convert.ToDouble(listBox8.Items[i])));
    listBox14.Items.Add(errory);
}
```

Рисунок 4.7 – Код розрахунку стандартних похибок вимірювання

Для того щоб отримати дані з листа під назвою «Items» в документі Excel, потрібно натиснути кнопку «Отримати дані». Результат наведено на рис. 4.8.

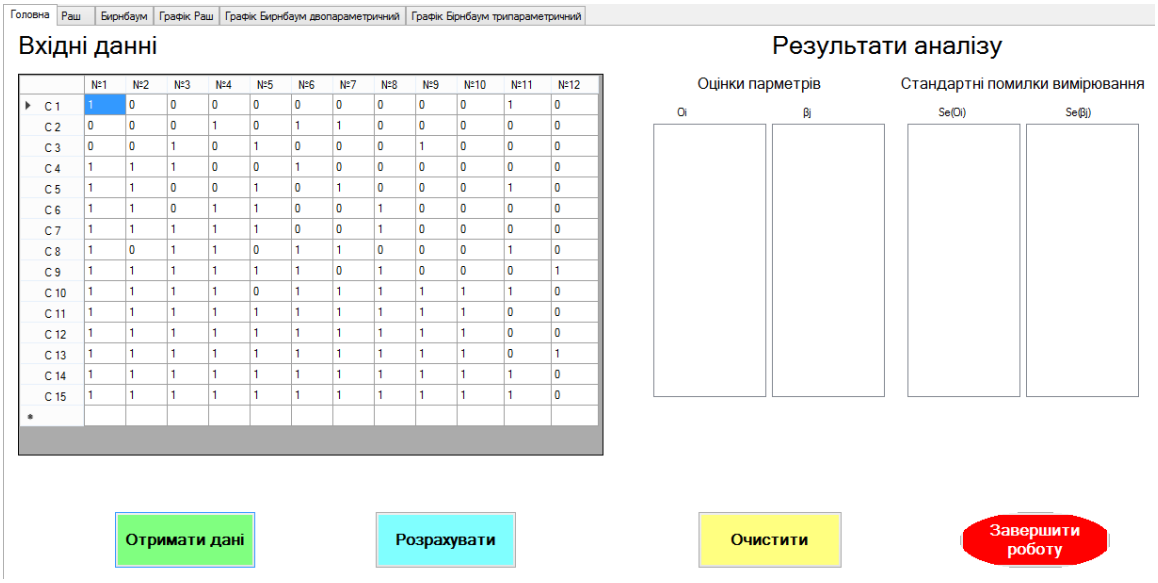

Рисунок 4.8 – Головна сторінка розробленого програмного забезпечення

Для підрахунку усіх коефіцієнтів необхідно натиснути кнопку «Розрахувати». Початкова таблиця даних і отримані розрахунки оцінок параметрів рівня підготовки θ і складності завдань β в єдиній інтервальній шкалі, а також стандартна похибка вимірювання $S(\theta_i)$, $S(\beta_j)$ показані у вікні на головній сторінці (рис. 4.9).

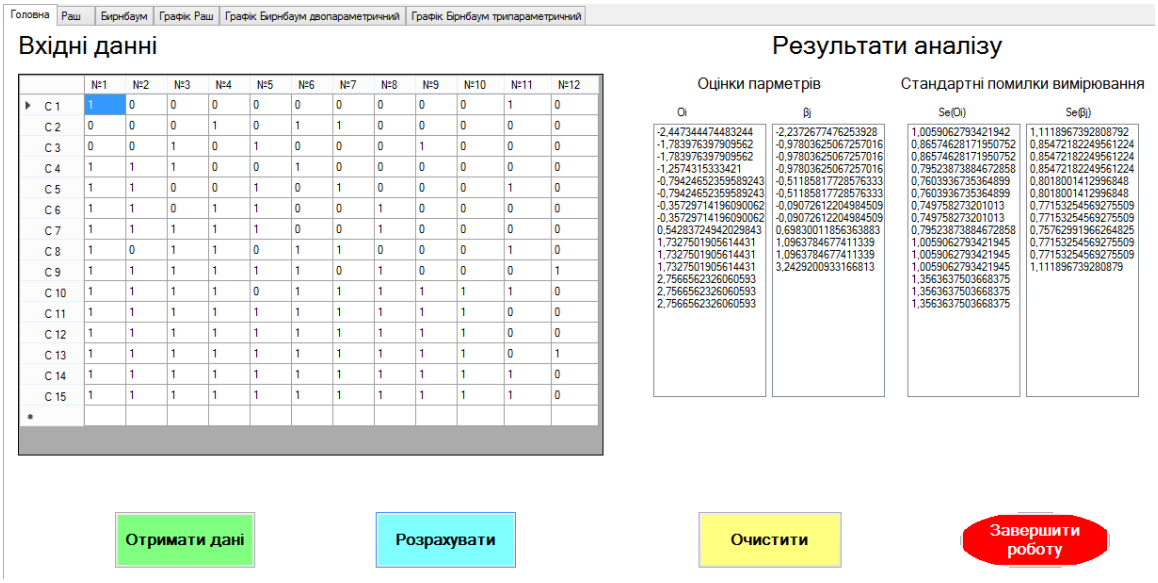

Рисунок 4.9 – Головна сторінка розробленого програмного забезпечення з відповідними розрахунками

Всі допоміжні розрахунки долей вірних і долей невірних відповідей i-го студента, долей вірних і долей невірних відповідей на кожне завдання тесту; початкова оцінка значень параметра, що характеризує рівень підготовки студентів і складності завдань тесту; середнє значення логітів рівня підготовки і логітів складності завдань тесту, а також дисперсії для множин θ_i^0 і β_j^0 наведено на рисунку 4.10.

Рисунок 4.10 – Сторінка для додаткових розрахунків по 1PL розробленого програмного забезпечення

Після підрахунку значень параметрів θ та β на шкалі логітів будують ансамбль характеристичних кривих завдань тесту. Для їх побудови складність β вважають параметром, а θ – незалежною змінною, значення якої вибирається довільно. Для даного випадку я обрала $\theta \in [-5,5]$, тому нулем у рамках даної теорії обирається 6 балів, так як в тесті відсутні результати 0 та 12 балів.

Ординати характеристичних кривих – значення функції P_j (ймовірність виконання завдання випробовуваним із рівнем підготовки θ), які підраховуються за формулою (2.1).

$$P_j(\theta) = \frac{e^{1,7(\theta-\beta_j)}}{1 + e^{1,7(\theta-\beta_j)}}.$$

Реалізація методу відбувається за допомогою коду (рис. 4.11). Будується сам графік і налаштовуються осі графіка.

```
double[] x = new double[21];
double[,] y = new double[21, 12];
double a = -5;
for (int i = 0; i < 21; i++)
{
    x[i] = a;
    a = a + 0.5;
}
for (int j = 0; j < listBox9.Items.Count; j++)
{
    double beta = Convert.ToDouble(listBox9.Items[j]);
    for (int i = 0; i < 21; i++)
    {
        double z = Math.Exp(1.7 * (x[i] - beta));
        y[i, j] = z / (1 + z);
    }
}

chart1.ChartAreas[0].AxisX.Minimum = 1;
chart1.ChartAreas[0].AxisX.Maximum = 21;

chart1.ChartAreas[0].AxisX.MajorGrid.Interval = 2.5;
for (int g = 0; g < 12; g++)
{
    Dictionary<string, double> chartsource = new Dictionary<string, double>();
    string graf = "Питання № " + Convert.ToString(g + 1);

    chart1.Series.Add(graf);
    chart1.Series[graf].BorderWidth = 5;

    for (int i = 0; i < 21; i++)
        chartsource.Add(x[i].ToString(), y[i, g]);

    chart1.Series[graf].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

    chart1.Series[graf].Points.DataBindXY(chartsource.Keys, chartsource.Values);
}
```

Рисунок 4.11 – Код побудови графіка ансамблю характеристичних кривих

Характеристичні криві 12 завдань, знайдені за допомогою формули однопараметричної моделі Г. Раша, наведені на рисунку 4.12.

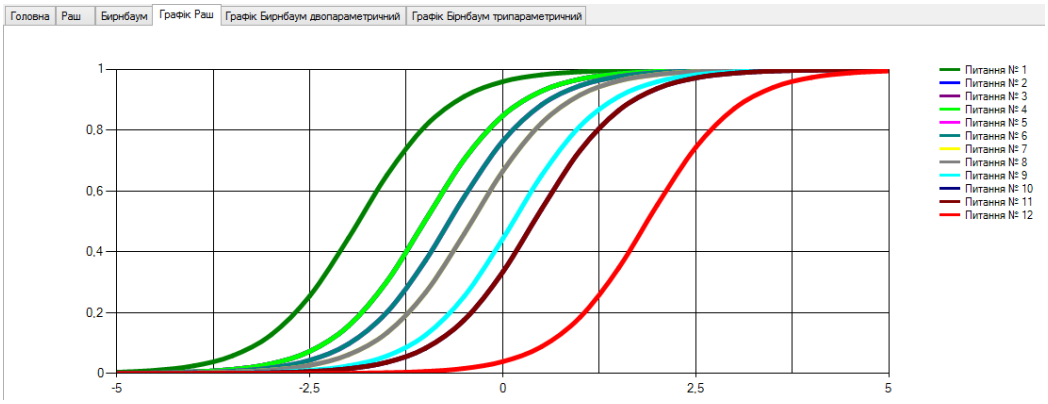

Рисунок 4.12 – Характеристичні криві 12 завдань тесту за 1PL

Аналогічно для реалізації двопараметричної моделі А. Бірнбаума завантажуються дані з листа під назвою «Items».

Для використання 2PL за формулою (2.3) потрібно знайти параметр диференційованої спроможності (дискримінативності) j-го завдання тесту за формулою: $a_j = \frac{(r_{bis})_j}{\sqrt{1-(r_{bis})_j^2}}$.

Для спрощення розрахунку коефіцієнта a_j використовуються точково-бісеріальний коефіцієнт, який знаходиться за формулою [4]:

$$\left(r_{pbis}\right)_j = \frac{(\bar{X}_1)_j - (\bar{X}_0)_j}{S_x} \cdot \sqrt{\frac{(N_1)_j \cdot (N_0)_j}{N(N-1)}}, \qquad (4.17)$$

де всі позначення ті ж що і для формули 2.6.

Перевагою використання точково-бісеріального коефіцієнту, замість бісеріального коефіцієнту, є простота в підрахунках і відсутність обов'язкових гіпотез, які висуваються в силу необхідності нормального характеру розподілу дихотомічних даних при знаходженні міри зв'язку.

Знайдемо коефіцієнти: $(\bar{X}_1)_j$ – середнє значення індивідуальних балів випробовуваних, які виконали вірно j-те завдання тесту, $(\bar{X}_0)_j$ – середнє значення індивідуальних балів випробовуваних, які виконали невірно j-те завдання тесту, $(N_1)_j$ – число випробовуваних, які виконали вірно j-те

завдання, $(N_0)_j$ – число випробовуваних, які виконали невірно j-те завдання, де N – загальна кількість випробовуваних. S_x – стандартне відхилення за множиною значень індивідуальних балів

Реалізація підрахунку точково-бісеріального коефіцієнту зображено на рисунку 4.13.

```
SeredVid();

double serx1 = 0;
double serx0 = 0;
for (int i = 0; i < dataGridView1.ColumnCount; i++)
{
    int countx1 = 0;
    int countx0 = 0;
    double sx1 = 0;
    double sx0 = 0;
    for (int j = 0; j < dataGridView1.RowCount - 1; j++)
    {
        if (Convert.ToInt32(dataGridView1.Rows[j].Cells[i].Value) == 1)
        {
            sx1 += Convert.ToDouble(listBox16.Items[j]);
            countx1 = countx1 + 1;
        }
        else
        {
            sx0 += Convert.ToDouble(listBox16.Items[j]);
            countx0++;
        }
    }

    serx1 = sx1 / countx1;
    listBox18.Items.Add(serx1);

    serx0 = sx0 / countx0;
    listBox19.Items.Add(serx0);
}


int n = 0;
for (int i = 0; i < listBox15.Items.Count; i++)
{
    n = Convert.ToInt32(listBox15.Items[i]) * (dataGridView1.RowCount - 1 - Convert.ToInt32(listBox15.Items[i]));
    listBox20.Items.Add(n);
}


textBox14.Text = Convert.ToString((dataGridView1.RowCount - 1) * ((dataGridView1.RowCount - 1) - 1));


double sqrt = 0;
for (int i = 0; i < listBox20.Items.Count; i++)
{
    sqrt = Math.Sqrt(Convert.ToDouble(listBox20.Items[i]) / Convert.ToDouble(textBox14.Text));
    listBox21.Items.Add(sqrt);
}


for (int i = 0; i < listBox18.Items.Count; i++)
{
    double x1x0sx = (Convert.ToDouble(listBox18.Items[i]) - Convert.ToDouble(listBox19.Items[i])) / Convert.ToDouble(textBox13.Text);
    listBox22.Items.Add(x1x0sx);
}


for (int i = 0; i < listBox22.Items.Count; i++)
{
    double rbis = Convert.ToDouble(listBox22.Items[i]) * Convert.ToDouble(listBox21.Items[i]);
    listBox23.Items.Add(rbis);
}
```

Рисунок 4.13 – Код розрахунку точково-бісеріального коефіцієнту

Розрахунок диференційованої спроможності (дискримінативності) j-го завдання тесту $a_j = \frac{(r_{bis})_j}{\sqrt{1-(r_{bis})_j^2}}$ реалізовано на рисунку 4.14.

```
for (int i = 0; i < listBox23.Items.Count; i++)
{
    double a_j = Convert.ToDouble(listBox23.Items[i]) / Math.Sqrt(1 - Math.Pow(Convert.ToDouble(listBox23.Items[i]), 2));
    listBox24.Items.Add(a_j);
}
```

Рисунок 4.14 – Код розрахунку дискримінативності j-го завдання тесту

Всі допоміжні коефіцієнти які були розраховані показані в програмі на сторінці Бірнбаум (рис. 4.15).

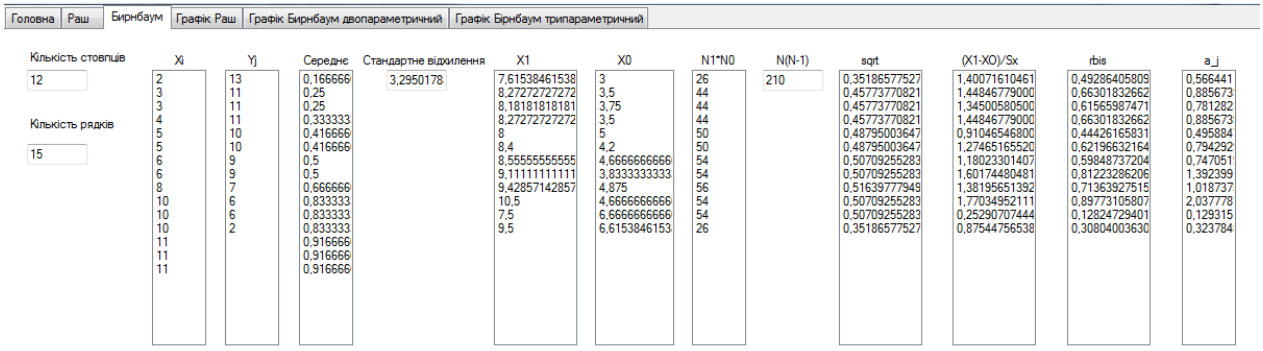

Рисунок 4.15 – Сторінка для додаткових розрахунків по 2PL розробленого програмного забезпечення

Після знаходження всіх параметрів в формулі (2.3) будуємо характеристичні криві завдань тесту. Для побудови характеристичної кривої завдання його складність β вважають параметром, а θ – незалежною змінною, значення якої вибирається довільно. Розглянемо параметр $\theta \in [-5,5]$, що відповідає шкалі оцінювання від 0 = «-5» до 10 = «5».

Ординати характеристичних кривих – значення функції P_j (ймовірність виконання завдання випробовуваним із рівнем підготовки θ), які підраховуються за формулою (2.3) :

$$P_j(\theta) = \frac{e^{1,7a_j(\theta-\beta_j)}}{1+e^{1,7a_j(\theta-\beta_j)}}.$$

Реалізація методу відбувається за допомогою коду на рисунку 4.16.

```
double[,] m = new double[21, r];
for (int j = 0; j < listBox12.Items.Count; j++)
{
    double beta = Convert.ToDouble(listBox12.Items[j]);
    double a_j = Convert.ToDouble(listBox24.Items[j]);
    for (int i = 0; i < 21; i++)
    {
        double z = Math.Exp(1.7 * a_j * (x[i] - beta));
        m[i, j] = z / (1 + z);
    }
}
```

Рисунок 4.16 – Код побудови графіка ансамблю характеристичних кривих

Характеристичні криві 12 завдань, знайдені за допомогою формули двопараметричної моделі А. Бірнбаума, наведені на рисунку 4.17.

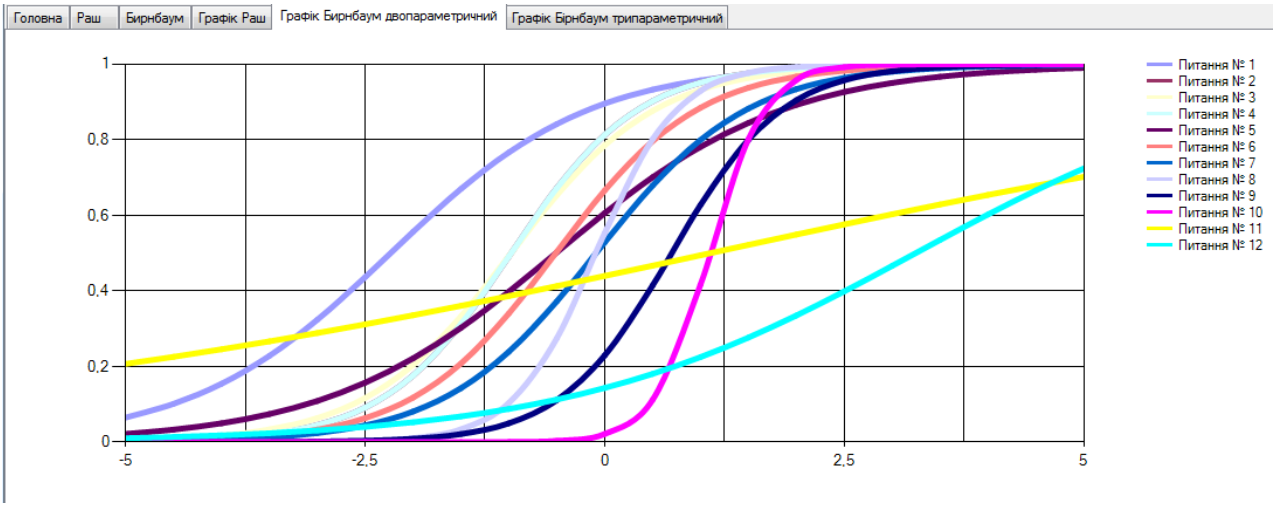

Рисунок 4.17 – Характеристичні криві 12 завдань тесту за 2PL

Аналогічно для реалізації трипараметричної моделі А. Бірнбаума завантажуються дані з листа під назвою «Items». Розраховуються всі коефіцієнти за описаним алгоритмом. Для використання 3PL за формулою (2.5) параметри $c_j = 0{,}25$.

$$P_j\{x_{ij} = 1 \mid \beta_j\} = 0{,}25 + 0{,}75 \cdot \frac{e^{1{,}7a_j(\theta-\beta_j)}}{1 + e^{1{,}7a_j(\theta-\beta_j)}}.$$

Реалізація методу відбувається за допомогою коду на рисунку 4.18.

```
double[,] t = new double[21, r];
for (int j = 0; j < listBox12.Items.Count; j++)
{
    double beta = Convert.ToDouble(listBox12.Items[j]);
    double a_j = Convert.ToDouble(listBox24.Items[j]);
    for (int i = 0; i < 21; i++)
    {
        double z = Math.Exp(1.7 * a_j * (x[i] - beta));
        t[i, j] = 0.25 + 0.75 * z / (1 + z);
    }
}
```

Рисунок 4.18 – Код побудови графіка ансамблю характеристичних кривих

Характеристичні криві 12 завдань, знайдені за допомогою формули трипараметричної моделі А. Бірнбаума, наведені на рисунку 4.19.

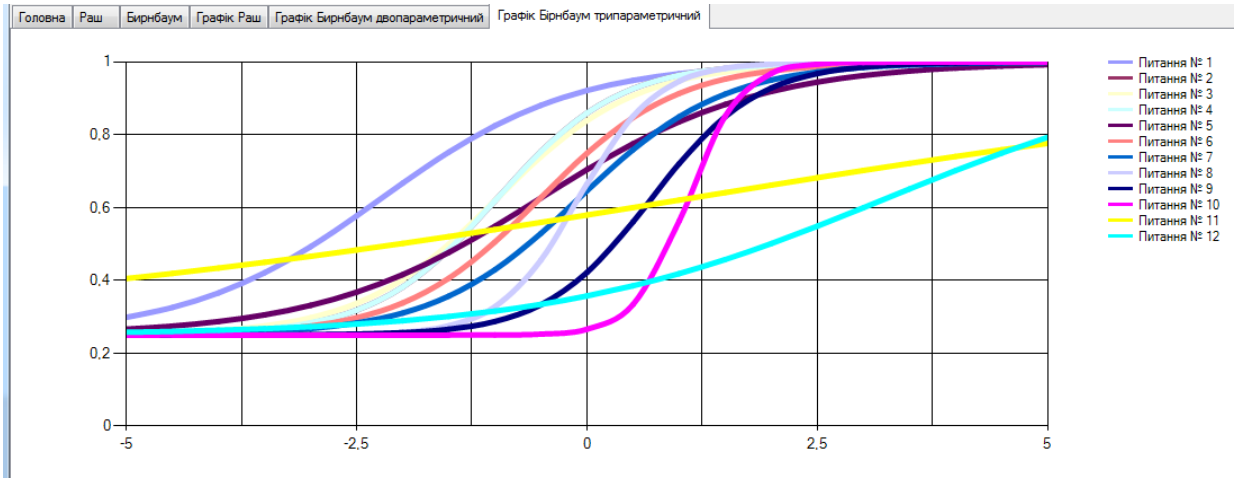

Рисунок 4.19 – Характеристичні криві 12 завдань тесту за 3PL

4.4 Висновки за 4 розділом

У даному розділі представлений алгоритм роботи з даним програмним забезпеченням, який надає можливість отримати коефіцієнти, що характеризують рівень підготовки випробовуваних і складності завдань тесту та графіки ансамблю характеристичних кривих 12 завдань тесту однопараметричною моделлю Г. Раша, двопараметричною і трипараметричною моделями А. Бірнбаума для подальшого аналізу якості педагогічного тесту.

ВИСНОВКИ

Розглянута теорія тестування і підходи для оцінювання якості тестування дозволяють зробити висновок про доцільність використання тестового підходу до оцінювання знань і перевірки якості тестування на основі моделі Item Response Theory (однопараметрична модель Г. Раша, двопараметрична та трипараметрична моделі А. Бірнбаума) для підвищення точності створення якісних і добре збалансованих тестів.

Зроблений ретельний аналіз існуючих програм для процедури статистичної обробки результатів тестування знань (FastTest, Xcalibre, Winsteps, ConQuest, LOGIST, RUMM).

Розроблено програмне забезпечення, що дозволяє отримати коефіцієнти оцінки латентних параметрів випробовуваних і параметрів завдань тесту та побудови ансамблю характеристичних кривих однопараметричної моделі Г. Раша, двопараметричної та трипараметричної моделей А. Бірнбаума для аналізу якості тестування.

ПЕРЕЛІК ПОСИЛАНЬ

1. Кухар Л. О., Сергієнко В. П. Конструювання тестів. Курс лекцій : навч. посіб. Луцьк : Видавець ПП Гадяк Ж.В., 2010. 182 с.

2. Булах І. Є., Мруга М. Р. Створюємо якісний тест : навчальний посібник. Київ : Майстер-клас, 2006. 160 с.

3. Беспалько В. П. Слагаемые педагогической технологии. Москва : Педагогика, 1989. 192 с.

4. Челышкова М. Б., Савельев Б. А. Методические рекомендации по разработке педагогических тестов для комплексной оценки подготовленности студентов в вузе. Москва : ИЦПКПС, 1995. 77 с.

5. Аванесов В. С. Основы научной организации педагогического контроля в высшей школе. Москва : МИСиС, 1989. 167 с.

6. Васьківська Г. О. Використання навчальних текстів у процесі формування системи знань про людину в учнів старшої школи // *Рідна школа*. 2013. № 6. С.41–46.

7. Звонников В. И., Челышкова М. Б. Современные средства оценивания результатов обучения : учеб. пособие для студ. высш. учеб. заведений. Москва : издательский центр «Академия». 2019. 224 с.

8. Майоров А. Н. Теория и практика создания тестов для системы образования. Москва : Народное образование, 2000. 352 с.

9. Лутченко Л. І., Пасічник Н. О. Основи педагогічного оцінювання: Навчально-методичний посібник. Кіровоград : Лисенко В. Ф., 2012. 72 с.

10. Авраменко О. В. Вимірювання в освіті : Підручник. Кіровоград : Лисенко В. Ф., 2011. 360 с.

11. Канівець Т. М. Основи педагогічного оцінювання : [навчально-методичний посібник]. Ніжин : Видавець ПП Лисенко М. М., 2012. 102 с.

12. Фетісов В. С. Комп’ютерні технології в тестуванні: навч.-метод. посіб. Ніжин : Видавець ПП Лисенко М. М., 2011. 140 с.

13. Анастази А., Урбина С. Психологическое тестирование. Санкт-Петербург : Питер. 2006. 688 с

14. Ландар І. Л. Валідність результатів вимірювання // *Інформаційно-комунікаційнй технології*. 2014. №1. С.35–36.

15. Аванесов В. С. Композиция тестовых заданий. Учебная книга для преподавателей вузов, учителей школ, аспирантов и студентов педвузов. 2 изд., испр. и доп. Москва : Адепт, 1998. 217 с.

16. Клайн П. Справочное руководство по конструированию тестов: Введение в психометрическое проектирование: Перевод с английского / Под ред. Л. Ф. Бурлачука. Киев : ПАН Лтд., 1994. 288 с.

17. Gulliksen H. Theory of Mental Tests. New York : Wiley, 1950. 486 p.

18. Guttman L. A special review of Harold Gulliksen, Theory of Mental Tests // *Psychometrika*, 1953. №18. pp. 133–130.

19. Guttman L. A basis for analyzing test-retest reliability // *Psychometrika*, 1945. №10. pp. 255–282.

20. Zimmerman D. W., Williams R. H., Zumbo B. D. Louis Guttman's Contributions to Classical Test Theory // *International Journal of Testing*, 2005. №5. pp. 81–95.

21. Lord F. M., Novick R. M. Statistical theories of mental test scores. Massachusetts : Addison-Wesley, 1986. 192 p.

22. Crocker L., Algina J. Introduction to classical and modern test theory. New York : Holt, Rinehart and Winston, 1986. 218 p.

23. Fisher R.A. Statistical Methods for Research Workers. London : Oliver and Boyd, 1925. 276 p.

24. Shavelson R. J., Webb N. M. Generalizability Theory : A Primer // *Journal of Educational Measurement*, 1993. №3. pp. 269–272.

25. Cronbach L. J., Nageswari R., Gleser, G. C. Theory of generalizability: A liberation of reliability theory // *The British Journal of Statistical Psychology*, 1963. №16, pp. 137–163.

26. Cronbach L. J., Gleser G. C., Nanda H., Rajaratnam N. The dependability of behavioral measurements: Theory of generalizability for scores and profiles. New York : John Wiley, 1972. 410 p.

27. Brennan R. L. Generalizability theory // *Educational Measurement*: *Issues and Practice*, 1992. № 11(4), pp. 27–34.

28. Brennan R. L. Performance assessments from the perspective of generalizability theory // *Applied Psychological Measurement*, 2000. №24 (4), pp. 339–353.

29. Brennan R. L. Generalizability theory and classical test theory // *Applied Measurement in Education*, 2011. №24(1), pp. 1–21.

30. Huang J. Generalizability theory as evidence of concerns about fairness in large-scale ESL writing assessments // *TESOL Journal*, 2011. № 2(4), pp. 423–443.

31. Huang J. Using generalizability theory to examine the accuracy and validity of large-scale ESL writing assessment // *Assessing Writing*, 2012. № 17(3), pp. 123–139.

32. Solano-Flores G., Li M. Generalizability theory and the fair and valid assessment of linguistic minorities // *Educational Research and Evaluation*, 2013. №19, pp. 245–263.

33. Rasch G. Probabilistic Models for Some Intelligence and Attainment Tests. Chicago : Univ. of Chicago Press, 1980. 199 p.

34. Birnbaum A. Some Latent Trait Models and Their Use in Inferring an Examinee's Ability. In Lord F. M. and Novick M. // *Statistical Theories of Test scores*. 1968. pp. 397–479.

35. Lord F. M. Application of Item Response Theory to Practical Testing Problems. Hillsdale. New Jersey : Lawrence Erlbaum Ass. 1980. 266 p.

36. Авраменко О. В. Вимірювання в освіті : Підручник. Кіровоград : Лисенко В. Ф., 2011. 360 с.

37. Лісова Т. В. Моделі та методи сучасної теорії тестів : [навчально-методичний посібник]. Ніжин : Видавець ПП Лисенко М. М., 2012. 112 с.

38. Аванесов В. С. Метрическая система Георга Раша – Rasch Measurement (RM) // *Педагогические Измерения*. 2010. № 2. С.3–30.

39. Аванесов В.С. Item Response Theory: Основные понятия и положения. Статья первая // *Педагогические Измерения*. 2007. № 2. С. 3–28.

40. Аванесов В.С. Истоки и основные понятия математической теории педагогических измерений (Item Response Theory). Статья вторая // *Педагогические Измерения*. 2007. № 3. С. 3–36.

41. Челышкова М. Б. Теория и практика конструирования педагогических тестов : Учебное пособие. Москва : Логос, 2002. 432 с.

42. Ким В. С. Тестирование учебных достижений. Монография. Уссурийск : Издательство УГПИ, 2007. 214 с.

43. Нейман Ю. М., Хлебников В. А. Введение в теорию моделирования и параметризации педагогических тестов. Москва : Прометей, 2000. 168 с.

44. Свиридов А. П. Стандартизированные методы на примере контроля и диагностирования знаний: монография. Москва : Издательство РГСУ, 2011. 294 с.

45. Самолюк Н. Г. Современные средства оценивания результатов обучения. Конспекты лекций. Томск : Томский государственный педагогический университет, 2015. 518 с.

46. Kingsbury G. G. Adapting Adaptive Testing: Using the MicroCAT Testing System in a Local School District. Educational Measurement: Issues and Practice, 1990. № 9(2), pp. 3–6.

47. Дудко А. Ф. Комп’ютерно орієнтована методика оцінювання якості тестів з вищої математики викладачами закладів вищої освіти : дис. на здобуття наук. ступеня канд. педаг. наук : 13.00.10 / Дудко Анна Федорівна. Київ, 2019. 293 с.

48. Wright B. D., Mead R. J., Bell S. R.. BICAL. Calibrating Items with the Rasch Model. *Statistical Laboratory, Department of Education. University of Chicago*. 1979. pp. 159–165.

49. Bock R. D., Aitkin M. Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm. // *Psychometrika*. 1981. №46. pp. 443–459.

50. Dempster A. P., Laird N. M., Rubin D. B. Maximum likelihood from incomplete data via the EM algorithm // *Journal of the Royal Statistical Society*. 1977. Vol. 39 (1). pp. 1–38.

51. Linden W. J., Hambleton R. K. Bock R. D., Zimowski M. F Multiple group IRT. *Handbook of modern item response theory*. New York : Springer-Verlag, 1997. pp. 433–448.

52. Weiss D. J., Minden S. A Comparison of Item Parameter Estimates from Xcalibre 4.1 and Bilog-MG (Technical Report). Minneapolis : Assessment Systems Corporation, 2012. 16 c.

53. Linacre J. M. Auser’s guide and manual to WINSTEPS: Rasch model computer programs. Chicago : Linacre, 2004. 213 p.

54. Mislevy R. J., Stocking M. L. A Consumer's Guide to LOGIST and BILOG // *Applied Psychological measurement*. 1989. Vol. 13, №. 1. pp. 57–75

55. Ким В. С. Обработка результатов тестирования компьютерной программой RUMM-2020 // *Педагогические Измерения*. 2008. №4. С.53–69.

56. Ким В. С. Анализ результатов тестирования в Rasch Measurement // *Педагогические Измерения*. 2005. №4. С. 39–45.

57. Смирнова Г. И. Алгоритм обработки матриц результатов тестирования с оценкой 0-1-2 и более с помощью программы RUMM-2010 // *Педагогические Измерения*. 2007. №4. С. 86–90.

ДОДАТОК А

Текст програми

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.OleDb;
using System.Windows.Forms.DataVisualization.Charting;
using System.IO;

namespace WindowsFormsApplication4
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();

            System.Drawing.Drawing2D.GraphicsPath myPath = new
System.Drawing.Drawing2D.GraphicsPath();
            myPath.AddEllipse(0, 0, button1.Width, button1.Height);
            Region myRegion = new Region(myPath);
            button1.Region = myRegion;
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void dataGridView1_CellContentClick(object sender,
DataGridViewCellEventArgs e)
        {

        }
        private void button1_Click(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            DanyDel();
            Open();
            Errors();
        }
        private void button3_Click(object sender, EventArgs e)
        {
            DanyDel();
            Rash();
            Birnbaum();
            Grafik();
```

```
        }
        private void button4_Click(object sender, EventArgs e)
        {
            DanyDel();
            dataGridView1.DataSource = null;
        }

        private void DanyDel()
        {
            foreach (var ctrl in tabPage2.Controls)
                if (ctrl is TextBox) (ctrl as TextBox).Clear();
            foreach (var ctrl in tabPage4.Controls)
                if (ctrl is TextBox) (ctrl as TextBox).Clear();
            foreach (Control x in tabPage1.Controls)
                if (x is ListBox)  ((ListBox)x).Items.Clear();
            foreach (Control x in tabPage2.Controls)
                if (x is ListBox) ((ListBox)x).Items.Clear();
            foreach (Control x in tabPage4.Controls)
                if (x is ListBox) ((ListBox)x).Items.Clear();
            chart1.Series.Clear();
            chart2.Series.Clear();
            chart3.Series.Clear();
        }

        private void Open()
        {
            dataGridView1.DataSource = null;
            try
            {
                OpenFileDialog ofd = new OpenFileDialog();
                ofd.InitialDirectory = "d:\\";
                ofd.DefaultExt = "*.xls;*.xlsx";
                ofd.Filter = "Microsoft Excel (*.xls*)|*.xls*";
                ofd.Title = "Выберите документ для загрузки данных";
                if (ofd.ShowDialog() != DialogResult.OK)
                {
                    MessageBox.Show("Вы не выбрали файл для открытия", "Загрузка
данных...", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                String constr = "Provider=Microsoft.ACE.OLEDB.12.0;Data Source="
+
                                    ofd.FileName + ";Extended Properties='Excel
12.0 XML;HDR=NO;IMEX=1';";

                System.Data.OleDb.OleDbConnection con = new
System.Data.OleDb.OleDbConnection(constr);
                con.Open();
                DataSet ds = new DataSet();
                DataTable schemaTable =
con.GetOleDbSchemaTable(System.Data.OleDb.OleDbSchemaGuid.Tables, new object[] {
null, null, null, "TABLE" });
                string sheet1 = (string)schemaTable.Rows[0].ItemArray[2];
                string select = String.Format("SELECT * FROM [{0}]", sheet1);
                System.Data.OleDb.OleDbDataAdapter ad = new
System.Data.OleDb.OleDbDataAdapter(select, con);
                ad.Fill(ds);
                DataTable dt = ds.Tables[0];
                con.Close();
                con.Dispose();
                dataGridView1.DataSource = dt;
```

```
                dataGridView1.AutoResizeColumns();
                    dataGridView1.RowHeadersWidth = 70;
                    for (int i = 0; i < dataGridView1.Columns.Count; i++)
                        dataGridView1.Columns[i].HeaderText = "№" +
Convert.ToString(i + 1);
                    for (int i = 0; i < dataGridView1.Rows.Count - 1; i++)
                        dataGridView1.Rows[i].HeaderCell.Value = "С " +
Convert.ToString(i + 1);
            }
            catch
            {
                dataGridView1.DataSource = null;
                MessageBox.Show("ПОМИЛКА !!!");
                MessageBox.Show("Виконайте вимоги до БАЗИ ДАНИХ");
            }
        }

        private double SummaColums(int b)
        {
            double sum = 0;
            for (int j = 0; j < dataGridView1.RowCount - 1; j++)
                sum += Convert.ToDouble(dataGridView1.Rows[j].Cells[b].Value);
            return sum;
        }

        private double SummaRows(int b)
        {
            double sum = 0;
            for (int j = 0; j < dataGridView1.ColumnCount; j++)
                sum += Convert.ToDouble(dataGridView1.Rows[b].Cells[j].Value);
            return sum;
        }

        private double Srednee(int b)
        {
            double sred = 0;
            double sum = SummaRows(b);
            double s = dataGridView1.ColumnCount;
            sred = sum / s;
            return sred;
        }

        private void SeredVid()
        {
            double srednpravxi = 0;
            double rizn = 0;
            double std = 0;
            for (int i = 0; i < listBox16.Items.Count; i++)
                srednpravxi += Convert.ToDouble(listBox16.Items[i]);

            double sred = srednpravxi / listBox16.Items.Count;
            for (int i = 0; i < listBox16.Items.Count; i++)
                rizn += Math.Pow((Convert.ToDouble(listBox16.Items[i]) - sred),
2);

            std = Math.Sqrt(rizn / (listBox16.Items.Count - 1));
            textBox13.Text = Convert.ToString(std);
        }

        private void Errors()
        {
            for (int i = 0; i < dataGridView1.ColumnCount; i++)
            {
```

```
                for (int j = 0; j < dataGridView1.RowCount - 1; j++)
                {
                    if (Convert.ToString(dataGridView1.Rows[j].Cells[i].Value)
!= "1")
                    {
                        if
(Convert.ToString(dataGridView1.Rows[j].Cells[i].Value) != "0")
                        {
                            MessageBox.Show("Помилкові дані !!!");
                            MessageBox.Show("Виконайте вимоги до БАЗИ ДАНИХ");
                            dataGridView1.DataSource = null;
                        }
                    }
                }
            }
        }

        private void Rash()
        {
            textBox1.Text = Convert.ToString(dataGridView1.ColumnCount);
            textBox2.Text = Convert.ToString(dataGridView1.RowCount - 1);

            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
listBox1.Items.Add(SummaRows(i));

            for (int i = 0; i < listBox1.Items.Count; i++)
            {
                double pi = Convert.ToDouble(listBox1.Items[i]) /
dataGridView1.ColumnCount;
                listBox2.Items.Add(pi);
            }

            for (int i = 0; i < listBox2.Items.Count; i++)
            {
                double qi = 1 - Convert.ToDouble(listBox2.Items[i]);
                listBox3.Items.Add(qi);
            }

            double sumoi = 0;
            for (int i = 0; i < listBox3.Items.Count; i++)
            {
                double oi = Math.Log(Convert.ToDouble(listBox2.Items[i]) /
Convert.ToDouble(listBox3.Items[i]));
                listBox4.Items.Add(oi);
                sumoi += oi;
            }

            double sumoi2 = 0;
            for (int i = 0; i < listBox4.Items.Count; i++)
            {
                double oi2 = Math.Pow(Convert.ToDouble(listBox4.Items[i]), 2);
                listBox5.Items.Add(oi2);
                sumoi2 += oi2;
            }
            textBox3.Text = Convert.ToString(sumoi2);

            double oo = sumoi / listBox4.Items.Count;
            textBox4.Text = Convert.ToString(oo);

            double vdispersia = (sumoi2 - (listBox5.Items.Count * Math.Pow(oo,
2))) / (listBox5.Items.Count - 1);
            textBox5.Text = Convert.ToString(vdispersia);
```

```
            for (int i = 0; i < dataGridView1.ColumnCount; i++)
                listBox6.Items.Add(SummaColums(i));

            for (int i = 0; i < listBox6.Items.Count; i++)
            {
                double pj = Convert.ToDouble(listBox6.Items[i]) /
(dataGridView1.RowCount - 1);
                listBox7.Items.Add(pj);
            }

            for (int i = 0; i < listBox7.Items.Count; i++)
            {
                double qi = 1 - Convert.ToDouble(listBox7.Items[i]);
                listBox8.Items.Add(qi);
            }

            double sumoj = 0;
            for (int i = 0; i < listBox8.Items.Count; i++)
            {
                double oj = Math.Log(Convert.ToDouble(listBox8.Items[i]) /
Convert.ToDouble(listBox7.Items[i]));
                listBox9.Items.Add(oj);
                sumoj += oj;
            }

            double sumoj2 = 0;
            for (int i = 0; i < listBox9.Items.Count; i++)
            {
                double oj2 = Math.Pow(Convert.ToDouble(listBox9.Items[i]), 2);
                listBox10.Items.Add(oj2);
                sumoj2 += oj2;
            }
            textBox6.Text = Convert.ToString(sumoj2);

            double bo = sumoj / listBox8.Items.Count;
            textBox7.Text = Convert.ToString(bo);

            double udispersia = (sumoj2 - (listBox10.Items.Count * Math.Pow(bo,
2))) / (listBox10.Items.Count - 1);
            textBox8.Text = Convert.ToString(udispersia);

            double xkut = Math.Sqrt((1 + (udispersia / 2.89)) / (1 -
((vdispersia * udispersia) / 8.35)));
            textBox9.Text = Convert.ToString(xkut);

            double ykut = Math.Sqrt((1 + (vdispersia / 2.89)) / (1 -
((vdispersia * udispersia) / 8.35)));
            textBox10.Text = Convert.ToString(ykut);

            double paramx = 0;
            for (int i = 0; i < listBox4.Items.Count; i++)
            {
                paramx = (xkut * Convert.ToDouble(listBox4.Items[i])) + bo;
                listBox11.Items.Add(paramx);
            }

            double paramy;
            for (int i = 0; i < listBox9.Items.Count; i++)
            {
                paramy = (ykut * Convert.ToDouble(listBox9.Items[i])) + oo;
                listBox12.Items.Add(paramy);
            }
```

```
            double errorx;
            for (int i = 0; i < listBox2.Items.Count; i++)
            {
                errorx = xkut / (Math.Sqrt(dataGridView1.ColumnCount *
Convert.ToDouble(listBox2.Items[i]) * Convert.ToDouble(listBox3.Items[i])));
                listBox13.Items.Add(errorx);
            }

            double errory;
            for (int i = 0; i < listBox7.Items.Count; i++)
            {
                errory = ykut / (Math.Sqrt((dataGridView1.RowCount - 1) *
Convert.ToDouble(listBox7.Items[i]) * Convert.ToDouble(listBox8.Items[i])));
                listBox14.Items.Add(errory);
            }
        }

        private void Birnbaum()
        {
            textBox12.Text = Convert.ToString(dataGridView1.ColumnCount);
            textBox11.Text = Convert.ToString(dataGridView1.RowCount - 1);


            for (int i = 0; i < dataGridView1.RowCount - 1; i++)
            {
                listBox16.Items.Add(SummaRows(i));
                listBox17.Items.Add(Srednee(i));
            }

            for (int i = 0; i < dataGridView1.ColumnCount; i++)
                listBox15.Items.Add(SummaColums(i));

            SeredVid();

            double serx1 = 0;
            double serx0 = 0;
            for (int i = 0; i < dataGridView1.ColumnCount; i++)
            {
                int countx1 = 0;
                int countx0 = 0;
                double sx1 = 0;
                double sx0 = 0;
                for (int j = 0; j < dataGridView1.RowCount - 1; j++)
                {
                    if (Convert.ToInt32(dataGridView1.Rows[j].Cells[i].Value) ==
1)
                    {
                        sx1 += Convert.ToDouble(listBox16.Items[j]);
                        countx1 = countx1 + 1;
                    }
                    else
                    {
                        sx0 += Convert.ToDouble(listBox16.Items[j]);
                        countx0++;
                    }
                }
                serx1 = sx1 / countx1;
                listBox18.Items.Add(serx1);
                serx0 = sx0 / countx0;
                listBox19.Items.Add(serx0);
            }
```

```
            int n = 0;
            for (int i = 0; i < listBox15.Items.Count; i++)
            {
                n = Convert.ToInt32(listBox15.Items[i]) *
(dataGridView1.RowCount - 1 - Convert.ToInt32(listBox15.Items[i]));
                listBox20.Items.Add(n);
            }

            textBox14.Text = Convert.ToString((dataGridView1.RowCount - 1) *
((dataGridView1.RowCount - 1) - 1));

            double sqrt = 0;
            for (int i = 0; i < listBox20.Items.Count; i++)
            {
                sqrt = Math.Sqrt(Convert.ToDouble(listBox20.Items[i]) /
Convert.ToDouble(textBox14.Text));
                listBox21.Items.Add(sqrt);
            }

            for (int i = 0; i < listBox18.Items.Count; i++)
            {
                double x1x0sx = (Convert.ToDouble(listBox18.Items[i]) -
Convert.ToDouble(listBox19.Items[i])) / Convert.ToDouble(textBox13.Text);
                listBox22.Items.Add(x1x0sx);
            }

            for (int i = 0; i < listBox22.Items.Count; i++)
            {
                double rbis = Convert.ToDouble(listBox22.Items[i]) *
Convert.ToDouble(listBox21.Items[i]);
                listBox23.Items.Add(rbis);
            }

            for (int i = 0; i < listBox23.Items.Count; i++)
            {
                double a_j = Convert.ToDouble(listBox23.Items[i]) / Math.Sqrt(1
- Math.Pow(Convert.ToDouble(listBox23.Items[i]), 2));
                listBox24.Items.Add(a_j);
            }
        }

        private void Grafik()
        {
            int r = Convert.ToInt32(textBox1.Text);
            double[] x = new double[21];
            double[,] y = new double[21, r];
            double a = -5;
            for (int i = 0; i < 21; i++)
            {
                x[i] = a;
                a = a + 0.5;
            }
            for (int j = 0; j < listBox9.Items.Count; j++)
            {
                double beta = Convert.ToDouble(listBox9.Items[j]);
                for (int i = 0; i < 21; i++)
                {
                    double z = Math.Exp(1.7 * (x[i] - beta));
                    y[i, j] = z / (1 + z);
                }
            }

            chart1.ChartAreas[0].AxisX.Minimum = 1;
```

```
            chart1.ChartAreas[0].AxisX.Maximum = 21;
            chart1.ChartAreas[0].AxisX.MajorGrid.Interval = 2.5;
            for (int g = 0; g < r; g++)
            {
                Dictionary<string, double> chartsource = new Dictionary<string,
double>();
                string graf = "Питання № " + Convert.ToString(g + 1);

                chart1.Series.Add(graf);
                chart1.Series[graf].BorderWidth = 5;
                for (int i = 0; i < 21; i++)
                    chartsource.Add(x[i].ToString(), y[i, g]);

                chart1.Series[graf].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;

                chart1.Series[graf].Points.DataBindXY(chartsource.Keys,
chartsource.Values);
            }


            double[,] m = new double[21, r];
            for (int j = 0; j < listBox12.Items.Count; j++)
            {
                double beta = Convert.ToDouble(listBox12.Items[j]);
                double a_j = Convert.ToDouble(listBox24.Items[j]);
                for (int i = 0; i < 21; i++)
                {
                    double z = Math.Exp(1.7 * a_j * (x[i] - beta));
                    m[i, j] = z / (1 + z);
                }
            }

            chart2.ChartAreas[0].AxisX.Minimum = 1;
            chart2.ChartAreas[0].AxisX.Maximum = 21;
            chart2.ChartAreas[0].AxisX.MajorGrid.Interval = 2.5;
            for (int g = 0; g < r; g++)
            {
                Dictionary<string, double> chartsource = new Dictionary<string,
double>();
                string grafb = "Питання № " + Convert.ToString(g + 1);

                chart2.Series.Add(grafb);
                chart2.Series[grafb].BorderWidth = 5;
                for (int i = 0; i < 21; i++)
                    chartsource.Add(x[i].ToString(), m[i, g]);

                chart2.Series[grafb].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
                chart2.Series[grafb].Points.DataBindXY(chartsource.Keys,
chartsource.Values);
            }


            double[,] t = new double[21, r];
            for (int j = 0; j < listBox12.Items.Count; j++)
            {
                double beta = Convert.ToDouble(listBox12.Items[j]);
                double a_j = Convert.ToDouble(listBox24.Items[j]);
                for (int i = 0; i < 21; i++)
                {
                    double z = Math.Exp(1.7 * a_j * (x[i] - beta));
                    t[i, j] = 0.25 + 0.75 * z / (1 + z);
```

```
                }
            }

            chart3.ChartAreas[0].AxisX.Minimum = 1;
            chart3.ChartAreas[0].AxisX.Maximum = 21;
            chart3.ChartAreas[0].AxisX.MajorGrid.Interval = 2.5;
            for (int g = 0; g < r; g++)
            {
                Dictionary<string, double> chartsource = new Dictionary<string,
double>();
                string graft = "Питання № " + Convert.ToString(g + 1);

                chart3.Series.Add(graft);
                chart3.Series[graft].BorderWidth = 5;
                for (int i = 0; i < 21; i++)
                    chartsource.Add(x[i].ToString(), t[i, g]);

                chart3.Series[graft].ChartType =
System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Spline;
                chart3.Series[graft].Points.DataBindXY(chartsource.Keys,
chartsource.Values);
            }
        }
    }
}
```