ФИЗИКО-МАТЕМАТИЧЕСКИЕ НАУКИ

УДК 004.9;519.8;13.02

ПРИНЦИПЫ МОДЕЛИРОВАНИЯ СИСТЕМЫ АНАЛИТИЧЕСКОГО ТЕСТИРОВАНИЯ ЗНАНИЙ НА ОСНОВЕ СИСТЕМЫ КОМПЬЮТЕРНОЙ МАТЕМАТИКИ MAPLE

© Г.Р.Адиятуллина, Ю.Г.Игнатьев

Рассмотрены принципы моделирования системы аналитического тестирования знаний на основе Maplet-приложений в СКМ Maple.

Ключевые слова: математическое моделирование, системы компьютерной математики, система аналитического тестирования.

1. Идея аналитического тестирования

В последнее время при анализе компетентности, профессиональных способностей сотрудников, знаний учащихся все чаще применяется тестирование. В связи с этим актуальным становится вопрос адекватности тестирования уровню знаний. Стандартное жесткое тестирование мало эффективно для оценки глубины понимания материала тестируемыми, особенно при анализе знаний по физико-математическим предметам. Этим объясняется необходимость построения гибкой системы тестирования, которая позволила бы осуществлять более глубокое зондирование профессиональных качеств тестируемого.

Интеллектуальные программы типа пакетов компьютерной математики, которые могут проводить аналитические вычисления, пригодны для осуществления идеи аналитического тестирования [1]. Эта идея заключается в сравнении ответа, полученного тестируемым, и эталонного ответа, полученного средствами СКМ.

2. Сравнение ответов

Сравнение ответов происходит путем нахождения разности их формульных или числовых выражений. При этом ответ может быть представлен в одном из многочисленных эквивалентных выражений. Программа тестирования устанавливает эквивалентность выражений ответа тестируемого и эталонного ответа, полученного средствами СКМ. В случае, если разность равна нулю, ответ, полученный тестируемым, считается верным. Пусть *F(x1,x2, ...)* – ответ тестируемого, а *F0(x1,x2, ...)* – эталонный ответ, полученный средствами СКМ. Сравнение ответов происходит по алгоритму:

```
if simplify(F-F0)=0
then F – верный ответ
else F – неверный ответ
end if
```

Процедура *simplify* осуществляет упрощение выражения. Приведем пример реализации алгоритма в виде процедуры, созданной в СКМ Maple и содержащейся в пользовательской библиотеке *CheckResult*. В данной процедуре происходит проверка координат точек максимума. Параметрами процедуры являются исследуемая функция, переменная, координаты точек максимума, найденные тестируемым. Координаты задаются в виде упорядоченного двумерного списка *[[x1,y1],[x2,y2],...]*.

```
>CheckResult[CheckCoorMax]:=proc(g,x,Z)
local K,n,m,l,i,n1,Z1,V,j,V1:
```

Сначала находятся верные координаты точек максимума исследуемой функции и их количество.

```
K:=PartResearch[CoorMax](g,x):
n:=PartResearch[QuantMax](g,x):
```

Далее задаются необходимые переменные.

```
Z1:=convert(Z,Matrix):
n1:=LinearAlgebra[RowDimension](Z1):
V:=Matrix(1..n1,1..n):
V1:=Vector(1..n1):
```

В следующем цикле производится сравнение эталонных и найденных тестируемым координат:

```
if n>0 then
for i from 1 to n1 do
for j from 1 to n do
if (simplify(K[j,1]-Z1[i,1])=0) and (simplify(K[j,2]-Z1[i,2])=0)
then V[i,j]:=1:
else V[i,j]:=0:
fi: od: od:
```

Далее производится подсчет числа верно найденных координат и вывод результата:

```
l:=0:
for i from 1 to n1 do m:=0:
for j from 1 to n do
if V[i,j]=1 then m:=m+1:
fi: od:
```

if m>0 then V1[i]:=1: l:=l+1: else V1[i]:=0:
fi: od:
print(`Количество верно найденных координат=`,l);
else
print(`Количество точек максимума найдено неверно`);
fi: end proc:

3. Возможности СКМ Maple. Процедуры

Система компьютерной математики Maple обладает возможностями, необходимыми для создания комплекса программ для аналитического тестирования и самотестирования. Данная система позволяет формировать собственные процедуры.

СКМ Maple содержит большое количество встроенных команд, процедур и функций. Тем не менее они не покрывают всех потребностей пользователей. Кроме этого, в некоторых случаях встроенные функции дают неверный результат. Очень часто они несут лишнюю информацию, выдают неудобочитаемый результат, запрашивают много параметров. Часто встроенные функции являются закрытыми, то есть пользователь не может проверить, посмотреть ход решения, вычисления. Поэтому в таких случаях целесообразным является создание собственных процедур и функций. В этом случае пользователь может создать процедуру или функцию, выдающие ответ в нужном ему виде. Также пользователь будет иметь возможность исправлять процедуру или функцию, если ему это понадобится. Для создания функции используется следующая конструкция [2]:

Fname := (x, y, ...)->expr

После этого вызов функции осуществляется в виде *Fname(x, y, ...)*, где *(x, y, ...)* – список формальных параметров функции пользователя с именем *Fname*. Переменные, указанные в списке формальных параметров, являются локальными. При подстановке на их место фактических параметров они сохраняют их значения только в теле функции (*expr*). Описанные таким образом функции пользователя фактически являются процедурами-функциями с несколько упрощенной структурой.

Простейшая форма задания процедуры следующая:

Pname := proc(Параметры) local внутренние переменные: Тело процедуры: end proc:

4. Библиотеки пользовательских процедур как инструмент создания системы аналитического тестирования

Пользовательские процедуры и функции можно объединять в библиотеки процедур. Такие библиотеки предусматривают возможность сокрытия их содержания. Процедуры, содержащиеся в них, могут быть использованы наравне с основными процедурами, заложенными разработчиками математического пакета, и доступны любым пользователям. Библиотека пользовательских процедур задается следующим образом:

> restart:
Library := table():
Library[f1] := proc(Параметры) Тело процедуры
f1 end:
Library[f2] := proc(Параметры) Тело процедуры f2 end:
Library[f3] := proc(Параметры) Тело процедуры f3 end:
save(Library, `c:/Library.m`);

где *Library* – имя библиотеки, *f1*,*f2*,*f3* – процедуры, входящие в библиотеку. Команда *save* осуществляет сохранение библиотеки в файле.

5. Maplet-приложение как инструмент создания системы аналитического тестирования

Организовать работу с пользовательскими библиотеками можно с помощью пакета Maplets. Maplets – пользовательские программы, которые облегчают процесс ввода и восприятия информации с помощью диалогового окна, которое может функционировать без запуска программы Maple. Графический интерфейс Maplet Builder позволяет разрабатывать достаточно сложные Maplet-приложения. Имеется возможность с помощью мыши вставлять в окно проекта кнопки, элементы прокрутки, графические окна и другие элементы интерфейса. Кроме этого можно выбором из списка устанавливать свойства для каждого из элементов, использовать встроенные стили для настройки внешнего вида, использовать предварительный просмотр в процессе создания проекта. Также Maplet-приложения можно создавать программными средствами Maple без использования Maplet Builder.

Создание библиотек пользовательских процедур приводит к необходимости запоминания большого количества новых названий процедур и их параметров. Кроме этого необходимо знать порядок ввода этих параметров и диапазоны их значений, что затрудняет доступ к ним как преподавателей, так и студентов. Применение маплетов избавляет пользователя от необходимости запоминать большой объем достаточно сложной информации, тем самым уменьшая вероятность ошибки ввода данных. Кроме этого, с помощью пакета Maplets можно создавать окна, диалоговые окна и другие визуальные объекты, которые

помогут пользователю, незнакомому с тонкостями программы Maple, получать все преимущества от использования приложения.

6. Принципиальная схема системы аналитического тестирования

Разработанная нами система аналитического тестирования имеет следующую структуру (Рис.1). Имеется несколько специализированных пользовательских библиотек, которые взаимосвязаны между собой. Все они используются при работе Maplet-приложения. Библиотека *PartResearch* содержит процедуры, предназначенные для нахождения эталонного решения задачи. Эта библиотека предназначена для преподавателей. Она позволяет подготавливать индивидуальные задания для студентов и демонстрационные материалы для проведения занятий. Библиотека *CheckResult* содержит процедуры, предназначенные для проверки решений, полученных студентами. Эти процедуры обращаются к процедурам библиотеки *PartResearch* для нахождения эталонного решения, а затем проверяют правильность проверяемого решения путем нахождения разности двух решений. Данная библиотека предназначена для преподавателей и студентов. Преподавателям она помогает проверять выполненную студентами работу, а студентам – осуществлять самоконтроль в процессе решения индивидуальных заданий. Библиотека *Tasks* содержит индивидуальные задания для студентов. Данная библиотека наполняется преподавателем, а используется студентами. В будущем для накопления и хранения заданий планируется использование базы данных. Библиотека *MarkScale* предназначена для преподавателей. Она содержит процедуры, позволяющие определить вес задачи, задать шкалу оценивания, а также выставить студенту окончательную оценку. При работе процедур данной библиотеки происходит обращение к процедурам библиотек *PartResearch*, *CheckResult* и *Tasks*. Маплет обращается к той или иной библиотеке в зависимости от выбранной траектории. Планируется первоначально создать две траектории – преподавателей и студентов, которым соответствуют две подсистемы. Вход в подсистему преподавателя будет защищен паролем. Преподаватель будет иметь доступ к получению готовых решений, проверке решений студентов, а также к заданию шкалы оценивания и выставлению оценки. Студент будет иметь возможность получать индивидуальные задания, а также проверять свои результаты в процессе решения.

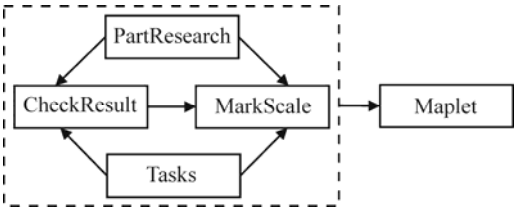

Рис.1. Блок-схема системы аналитического тестирования

7. Пример создания пользовательских библиотек и Maplet-приложения

Ранее в работе [3] была описана система аналитического тестирования, включающая несколько специализированных библиотек.

Рассмотрим подробнее библиотеку *PartResearch*. Каждая из процедур, входящих в данную библиотеку, представляет собой один шаг в исследовании функции. Все вместе они осуществляют полное аналитическое исследование функции действительного переменного и построение ее графика, а именно проверяют на четность, находят координаты пересечения графика функции с осью OX, находят промежутки знакового постоянства функции, находят количество точек минимума и максимума, координаты точек экстремума, промежутки монотонности функции, количество точек перегиба, их координаты, промежутки выпуклости и вогнутости, асимптоты, а также осуществляют автоматическое построение графика исследуемой функции в оптимальной области ее определения. Все процедуры соответствуют стандартной схеме исследования функции одной переменной, принятой в России [4]. Процедуры данной библиотеки используются затем в библиотеке *CheckResult*, а также при вычислении веса задачи. Библиотека предназначена для преподавателей.

[ExtremeCoorX, QuantExtreme, ExtremeCoorY, ExtremeCoorXY, QuantMin, QuantMax, CoorMin, CoorMax, SegmentMin, SegmentMax, CoorPosInflectX, QuantPosInflect, CoorPosInflectY, CoorPosInflectXY, QuantInflect, CoorInflect, ConcavInterval, MonotonyInterval, SignConstInterval, CrossOX, EvenOddFunc, TopY, BottomY, TopX, BottomX, DrawMax, DrawMin, DrawInflect, DrawAsymptote, Graph, CompleteResearch]

Библиотека *CheckResult* содержит процедуры, позволяющие проверить правильность результатов, полученных студентами в процессе решения, а именно проверка количества и координат точек максимума, точек минимума, точек перегиба. Библиотека предназначена для студентов и преподавателей. Используется при вычислении результата (оценки) студента, а также для самотестирования студентов.

[CheckQuantExtreme, CheckQuantMax, CheckQuantMin, CheckQuantInflect, CheckCoorMax1, CheckCoorMax, CheckCoorMin, CheckCoorInflect]

Библиотека *Tasks* содержит индивидуальные задания студентов по вариантам: *[T1, T2, T3, T4, T5, T6, T7, T8, T9, T10]* Кроме проверки решения, существует проблема оценки данного решения. Например, если студент нашел часть точек экстремума. В данном случае необходимо более гибко оценить работу студента с учетом веса правильно выполненной работы. Для этого разработана библиотека *MarkScale*, которая содержит процедуры определения веса задачи, задания шкалы оценивания, вычисления оценки.

Маплет играет роль связующего звена между библиотеками и пользователями (преподавателями, студентами). Рассмотрим процесс создания маплета. Работа над маплетом начинается с подключения специального пакета Maplet[Elements], который содержит процедуры, необходимые для работы маплета [5]. После этого необходимо осуществить чтение пользовательских библиотек.

```
> with(Maplets[Elements]):
> read(`./CheckResult.m`):
with(CheckResult):
> read(`./PartResearch.m`):
with(PartResearch):
> read(`./Tasks.m`):
with(Tasks):
```

Далее в программе идет описание отображаемых элементов маплета. Например, главное окно "*CAT*" маплета содержит пять кнопок, каждая из которых позволяет осуществить переход к другому окну маплета (окну "*Получение заданий*", окну "*Решение задачи*", окну "*Проверка решения задачи*", окну "*О программе*") или осуществить действие (завершить работу маплета) (рис.2).

```
Window['Win1']("CAT",
[Button("Получение заданий",
'onclick'=Action(RunWindow('WV'))),
Button("Решение задачи", 'on-
click'=Action(RunWindow('WR'))),
Button("Проверка решений задачи", 'on-
click'=Action(RunWindow('WP'))),
Button("О программе", 'on-
click'=Action(RunWindow('WI'))),
Button("Выход", Shutdown())] ), …
```

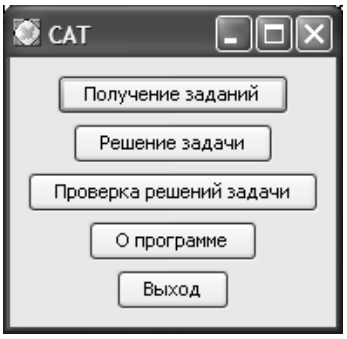

Рис.2. Главное окно "*CAT*"

В окне "*Решение задачи*" представлены на выбор шаги исследования функции (рис.3). Таким образом, можно найти только ту информацию, которая необходима на данный момент, – например, координаты точек максимума. Переход к нужному элементу задачи происходит при нажатии соответствующей кнопки.

```
Window['WR']("Решение задачи",
[Button("Координаты точек экстремума",
'onclick'=Action(RunWindow('WR2'))),
Button("Количество минимумов", 'on-
click'=Action(RunWindow('WR3'))),
Button("Количество максимумов", 'on-
click'=Action(RunWindow('WR4'))),
```

Рис.3. Окно "*Решение задачи*"

```
Button("Координаты минимумов", 'on-
click'=Action(RunWindow('WR5'))),
Button("Координаты максимумов", 'on-
click'=Action(RunWindow('WR6'))),
```

```
Button("Количество точек перегиба", 'on-
click'=Action(RunWindow('WR7'))),
Button("Координаты точек перегиба", 'on-
click'=Action(RunWindow('WR8'))),
Button("Промежутки выпуклости", 'on-
click'=Action(RunWindow('WR9a'))),
Button("Промежутки вогнутости", 'on-
click'=Action(RunWindow('WR9b'))),
Button("Промежутки монотонности (возр.)",
'onclick'=Action(RunWindow('WR10a'))),
Button("Промежутки монотонности
(убыв.)", 'onclick'=Action(RunWindow('WR10b'))),
Button("Промежутки знакопост-ва f(x)>0",
'onclick'=Action(RunWindow('WR11a'))),
Button("Промежутки знакопост-ва f(x)<0",
'onclick'=Action(RunWindow('WR11b'))),
Button("Нули функции", 'on-
click'=Action(RunWindow('WR12'))),
Button("Четность функции", 'on-
click'=Action(RunWindow('WR13'))),
Button("Выход", 'on-
click'=Action(CloseWindow('WR'))) ),
```

Рассмотрим одно из окон для нахождения готового решения на примере окна "*Координаты максимумов*" (рис.4). Здесь необходимо ввести исследуемую функцию и нажать кнопку "Вычислить". В поле "Координаты максимумов" появятся сами координаты.

```
Window['WR6']("Координаты максимумов",
[ [ "Введите f(x):",
TextField['TF61']('width'=30)],
[ "Координаты максимумов:",
TextBox['TF62']('width'=30, 'height'=10,
'font'=Font("Times new roman",italik,14))], [
Button("Вычислить",
Evalu-
ate('TF62'='convert(PartResearch[CoorMax](TF61,
x), 'listlist')')),
Button("Выход", 'on-
click'=Action(CloseWindow('WR6'))) ] ), ...
```

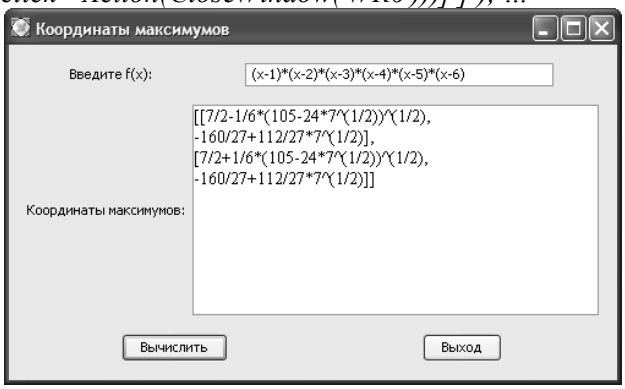

Рис.4. Окно "*Координаты максимумов*"

В окне "*Проверка решения задачи*" имеется возможность проверить полученные студентом результаты (рис.5). Каждый шаг исследования проверяется отдельно. Для перехода к нужному элементу нужно нажать на соответствующую кнопку.

```
Window['WP']("Проверка решения задачи",
[Button("Проверка количества точек макси-
мума",
'onclick'=Action(RunWindow('WP2'))),
Button("Проверка координат точек макси-
мума",
'onclick'=Action(RunWindow('WP3'))),
Button("Проверка количества точек миниму-
ма",
'onclick'=Action(RunWindow('WP4'))),
Button("Проверка координат точек миниму-
ма",
'onclick'=Action(RunWindow('WP5'))),
Button("Проверка количества точек переги-
ба",
'onclick'=Action(RunWindow('WP6'))),
Button("Проверка координат точек переги-
ба",
'onclick'=Action(RunWindow('WP7'))),
 Button("Выход",
'onclick'=Action(CloseWindow('WP'))) ), ...
```

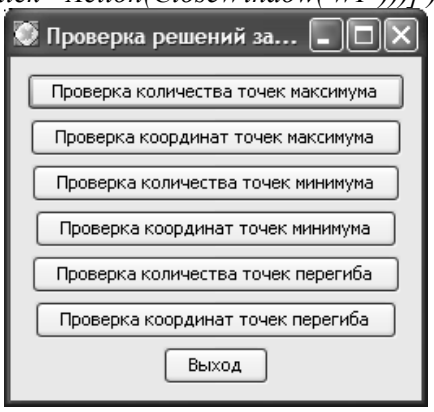

Рис.5. Окно "Проверка решения задачи"

Рассмотрим пример проверки координат точек максимума (рис.6). Для этого необходимо ввести исследуемую функцию, затем ввести полученные координаты и нажать кнопку "*Проверить*". В поле "*Ответ*" появится соответствующий результат.

```
Window['WP3']("Проверка координат точек
максимума (несколько пар)",[ [ "Функция:",
TextField['WP3TF1'] ('width'=30)],["Координаты
точек максимума:",
TextField['WP3TF2']('width'=20)],
["Ответ:", TextField['WP3TF3']('width'=30)],
[ Button("Проверить",Evaluate('WP3TF3'=
'CheckRe-
sult[CheckCoorMax](WP3TF1,x,WP3TF2)' )),
Button("Выход", 'on-
click'=Action(CloseWindow('WP3')))]])
```

В окне "*Получение заданий*" студенты смогут получить задания по вариантам (рис.7). Для того чтобы получить задание, необходимо ввести вариант и нажать кнопку "*Получить задание*".

```
Window['WV']("Получение заданий",
[ [ "Ваш вариант:",
TextField['WVTF1']('width'=5)],
[ Button("Получить задание", Evaluate('WVTB2'='Tasks[T](WVTF1)' ))],
[TextBox['WVTB2']('width'=20, 'height'=5,
'font'=Font("Times new roman",italik,14))] ] ),
...
```

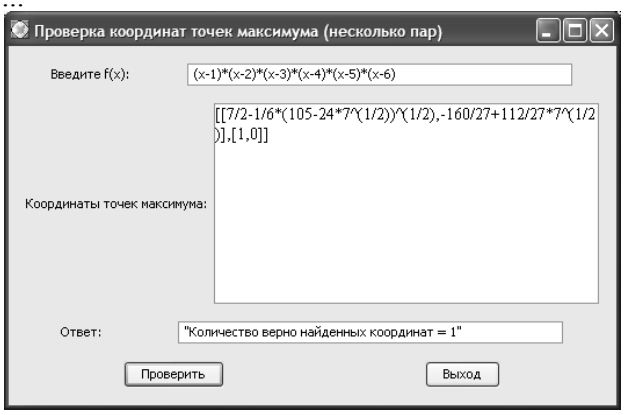

Рис.6. Окно "*Проверка координат точек максимума*"

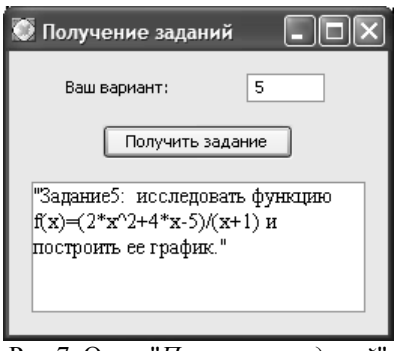

Рис.7. Окно "*Получение заданий*"

Кроме всего описанного, на главном окне предусмотрена инструкция пользователя, позволяющая разобраться, как работать с данным приложением. Для запуска маплета необходимо после описания содержимого маплета ввести оператор *Display*.

```
> Maplets[Display](maplet);
```

Работа над маплетом завершается сохранением файла с расширением .maplet. После этого можно запускать сохраненный файл, который уже будет работать без запуска Maple.

7. Оценка веса задачи, шкала оценивания

Оценка веса задачи осуществляется следующим образом. Задача разбивается на пункты – мелкие, промежуточные задачи. Затем вычисляется количество этих пунктов. Чем больше это число, тем больше вес задачи. Вес задачи необходимо вычислять для того, чтобы составлять эквивалентные (равные) по весу контрольные работы и наборы индивидуальных заданий. Вес каждого пункта выражается в процентах от веса самой задачи. Поэтому при проверке решения, полученного тестируемым, можно легко определить процент правильно найденных ответов пунктов. Благодаря процентному выражению веса пунктов решение, полученное тестируемым, можно оценить с использованием любой шкалы оценивания, которая может быть и пятибалльной, и 100-балльной, и любой другой. Таким же образом можно осуществлять оценку решения группы задач. В этом случае сами задачи будут выступать в роли пунктов.

Заключение

Таким образом, реализация системы аналитического тестирования знаний на базе Maplet-приложений может обогатить существующие системы тестирования следующими новыми характеристиками:

1. Более интуитивным интерфейсом, предоставляющим возможность начинающему пользователю быстро освоить систему тестирования;
2. Простотой использования и привлекательным графическим интерфейсом, позволяющими повысить мотивацию для работы с приложением как преподавателей, так и студентов;
3. Сокращением затрат времени преподавателей при подготовке и проверке заданий;
4. Интерактивностью приложения, обеспечивающей удобство и функциональность работы.

* * * * * * * * * *

1. *Игнатьев Ю.Г.* Использование аналитических возможностей пакета Maple для создания программ аналитического тестирования, самотестирования и генерации индивидуальных заданий в курсах высшей математики. Проблемы информационных технологий в математическом образовании: Учебное пособие / Под ред. Ю.Г.Игнатьева. – Казань: ТГППУ, 2005. – 118 с.
2. *Дьяконов В.П.* Maple 9.5/10 в математике, физике и образовании. – М.: СОЛОН-Пресс, 2006.
3. *Адиятуллина Г.Р.* Библиотеки пользовательских процедур в СКМ по курсу математического анализа: "Функции". Системы компьютерной математики и их приложения: материалы международной конференции. – Смоленск: СмолГу, 2009. – Вып.10. – 303 с.
4. *Фихтенгольц Г.М.* Курс дифференциального и интегрального исчисления. – М.: Физматлит, 2002. – Т.1. – 432 с.
5. *Кирсанов М.Н.* Maple 13 и Maplet. Решение задач механики. – М.: Физматлит, 2010.
6. *Адиятуллина Г.Р.* Маплеты как средство аналитического тестирования знаний по математическим курсам. Труды Математического центра имени

Н.И.Лобачевского: Материалы Восьмой молодежной научной школы-конференции "Лобачевские чтения-2009". – Казань: Казан. матем. об-во, 2009. – Т.39. – 417 с.

7. *Адиятуллина Г.Р.* Система аналитического тестирования в форме маплетов. Системы компьютерной математики и их приложения: материалы XI международной научной конференции, посвященной 70-летию профессора В.П.Дьяконова. – Смоленск: СмолГу, 2010. – Вып.11. – 342 с.

THE PRINCIPLES OF MODELING OF THE KNOWLEDGE ANALYTICAL TESTING SYSTEM ON THE BASIS OF COMPUTER MATHEMATICS PACKAGE MAPLE

G.R.Adiyatullina, Yu.G.Ignatyev

The principles of system modeling of analytic testing of knowledge on the basic of computers mathematics package maple are examined in the article.

Key words: mathematic modeling, computers mathematic systems, system of analytic testing.

* * * * * * * * * *

Адиятуллина Гульшат Рафисовна – соискатель кафедры геометрии и математического моделирования Татарского государственного гуманитарно-педагогического университета.

E-mail: gulshaton@mail.ru

Игнатьев Юрий Геннадиевич – доктор физико-математических наук, профессор, заведующий кафедрой геометрии и математического моделирования Татарского государственного гуманитарно-педагогического университета.

E-mail: ignatev_yu@rambler.ru