Московский Авиационный Институт
(национальный исследовательский университет)

Факультет Прикладной математики и физики

Кафедра Вычислительной математики и программирования

Курсовая работа

по дисциплине

«Численные методы»

IV курс, VII семестр

Студент: Белоусов С. Л.

Группа: 08-406

Руководитель: Ревизников Д. Л.

Москва 2013

Содержание

Общая информация

Данная работа состоит из 8 отдельных программ, которые представляют собой усовершенствованные версии лабораторных работ по курсу «Численные методы» и позволяют решать различные задачи линейной и нелинейной алгебры, математического анализа, ОДУ и УРЧП (подробные описания каждой из программ приведены далее). Все программы имеют графический интерфейс и выполнены в едином стиле. Встроенный парсер математических выражений позволяет использовать в качестве входных данных не только числовые значения, но и формулы. Полученные в ходе расчетов результаты отображаются в специальных полях в интерфейсе решателя, рисуются на графиках, а также сохраняются в отдельный файл. Поскольку составленные программы могут обрабатывать любой корректный ввод (в том числе все варианты заданий из лабораторных работ), они могут служить удобным примером для реализации собственных решателей, применяясь для сравнения результатов. Что касается технических деталей реализации, все программы написаны на языке C++ с использованием фреймворка Qt (для создания графического интерфейса) и дополнительной библиотеки Qwt (для построения графиков).

Запуск программ

Данная работа выполнена в двух версиях: для операционных систем семейства Microsoft Windows (как 32, так и 64-битных) и для 64-битных ОС семейства Linux. Каждая из версий содержится в отдельном каталоге с соответствующим названием. В сборке для Windows следует запускать исполняемый файл (.exe) с номером выбранной лабораторной, в случае с Linux нужно запустить одноименный bash-скрипт (.sh). При возникновении проблем с запуском рекомендуется почитать следующий абзац.

Важным моментом является тот факт, что все программы в данной работе используют динамическую линковку сторонних библиотек (это означает, что необходимые библиотеки не содержатся непосредственно в исполняемом файле, а ищутся в системе в момент запуска программы). От статической линковки в силу ряда причин было решено отказаться. Эта особенность несколько усложняет процесс запуска программ, поскольку требует наличия дополнительных файлов с динамическими библиотеками (такие файлы имеют расширение dll в Windows и so в Linux). Как правило, большинство данных файлов содержатся в самой операционной системе. Но некоторые из них, например, входящие в состав Qt, требуют отдельной установки. Необходимые файлы были по возможности включены в сборку, но гарантировать, что их всегда будет достаточно, к сожалению, нельзя. Для определения недостающих библиотек в Linux может оказаться полезной утилита ldd. В Windows вся необходимая информация обычно содержится в сообщении об ошибке.

Таким образом, в некоторых случаях может потребоваться найти и установить недостающие библиотеки, добавив их к уже имеющимся в сборке (папка libs в версии для Linux, директория с исполняемыми файлами в версии для Windows)

либо в одну из предназначенных для этого системных директорий.
Также могут возникнуть проблемы, если в системе уже установлены некоторые из содержащихся в сборке библиотек. В таком случае нужно удалить из сборки конфликтующие библиотеки.
Работа всех программ тестировалась в следующих ОС:

1. Ubuntu 12.04 LTS 64-bit
2. Ubuntu 13.10 64-bit
3. Windows 7 Professional 64-bit

Выходные файлы
Рассмотрим подробнее механизм работы с выходными файлами. Каждая из программ при выполнении расчетов создает файл, в который сохраняются полезные данные. Такие файлы бывают двух типов: лог-файлы и файлы с результатами. Основное различие между ними заключается в том, что лог-файлы содержат не сами результаты, а различные служебные данные из расчетов. В каждой программе используется только один тип файлов. Лог-файлы присутствуют в тех программах, где конечный результат занимает совсем небольшой объем и целиком размещается в полях интерфейса, но интерес также представляют дополнительные сведения о ходе расчета (например, результат LUP-разложения при решении СЛАУ соответствующим методом). В других же программах, напротив, результаты расчетов имеют значительные размеры, но помимо них какие-либо полезные данные отсутствуют. В таком случае полученные результаты просто сохраняются в выходной файл, а в интерфейсе они отображаются с помощью графиков.
Ниже приведена табличка, показывающая распределение типов выходных файлов по лабораторным работам.

Номер работы	1	2	3	4	5	6	7	8
Лог-файл	+	+	+					
Файл с результатами				+	+	+	+	+

Каждый выходной файл соответствует одному запуску программы и имеет название, состоящее из имени типа (log или res), за которым следует временная метка текущего запуска (с точностью до секунды). Все выходные файлы имеют расширение txt. Также стоит отметить, что лог-файл создается непосредственно в момент запуска программы и содержит сведения обо всех произведенных в данном запуске расчетах. В свою очередь, файл с результатами создается лишь при выполнении первого расчета и перезаписывается при каждом новом расчете в данном запуске.

Парсер математических выражений
Данный компонент присутствует во всех лабораторных, кроме первой (в ней ввод формул не требуется). Парсер является собственной разработкой, а не сторонней библиотекой, и устроен достаточно просто. Поддерживается

использование переменных, а также имеется набор встроенных функций и констант. Впрочем нужно отметить, что на вводимые выражения накладываются довольно строгие ограничения. В частности, в них нельзя пропускать операторы умножения (как это бывает принято в рукописной записи), а также следует избегать записи двух операторов подряд (например, +-), вместо этого разделяя их скобками. Более подробно о правилах записи выражений можно прочитать непосредственно в программе при нажатии кнопки «Справка». Там же приведен список встроенных функций и констант. Что касается реализации парсера, для вычисления выражений применяется их перевод в обратную польскую запись.

Прочие замечания

1. Существует удобная система навигации по полям и вкладкам интерфейса программы, не требующая управления мышью. Для перехода к следующему полю достаточно нажать клавишу TAB.
2. В целом ввод данных не требует использования каких-либо командных клавиш. Исключение составляет ввод аналитического решения (в тех лабораторных, где он имеется). По окончании ввода выражения для аналитического решения следует нажать клавишу Enter, чтобы изменения данного поля вступили в силу. Это сделано в целях удобства (вместо добавления отдельной кнопки). Также стоит отметить возможность использования клавишей со стрелками при вводе числовых данных, что позволяет быстро итерироваться по возможным значениям текущего поля.
3. В некоторых случаях на графиках могут «обрезаться» (быть не видны) левые края значений на оси ординат. Данная проблема решается перестроением графика без переключения вкладок (Ввод данных / Результат). Например, это можно сделать, если вырезать аналитическое решение, нажать Enter, вставить только что вырезанное решение и еще раз нажать Enter. В лабораторных, где графики строятся по срезам координат, для решения проблемы достаточно переключить текущий срез на другой, а затем обратно. Эта проблема связана с особенностями графической библиотеки Qwt и, по всей видимости, не существует подходящего для данных работ способа ее устранения.
4. Названия полей, являющиеся именами различных параметров и функций, как правило, имеют расшифровку. Она либо выдается при наведении курсора на название поля, либо присутствует в пояснительной картинке («Общий вид решаемой задачи»).

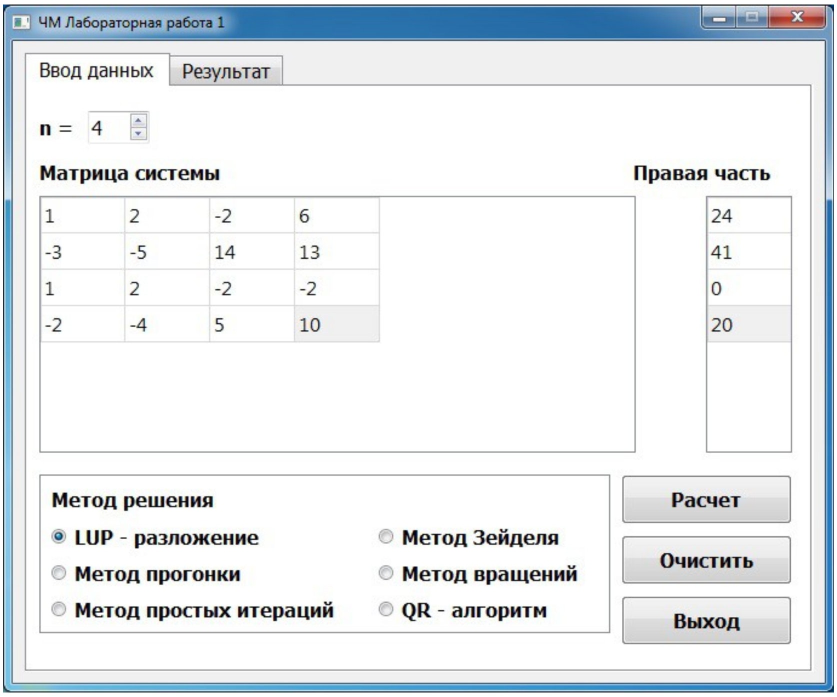
Лабораторная работа 1

Реализованный метод	Решаемая задача
LUP-разложение	Решение СЛАУ, нахождение обратной матрицы и определителя матрицы системы
Метод прогонки	Решение СЛАУ с трехдиагональной матрицей
Метод простых итераций	Решение СЛАУ
Метод Зейделя	Решение СЛАУ
Метод вращений (метод Якоби)	Нахождение собственных значений и собственных векторов симметричной матрицы
QR-алгоритм	Нахождение собственных значений произвольной матрицы

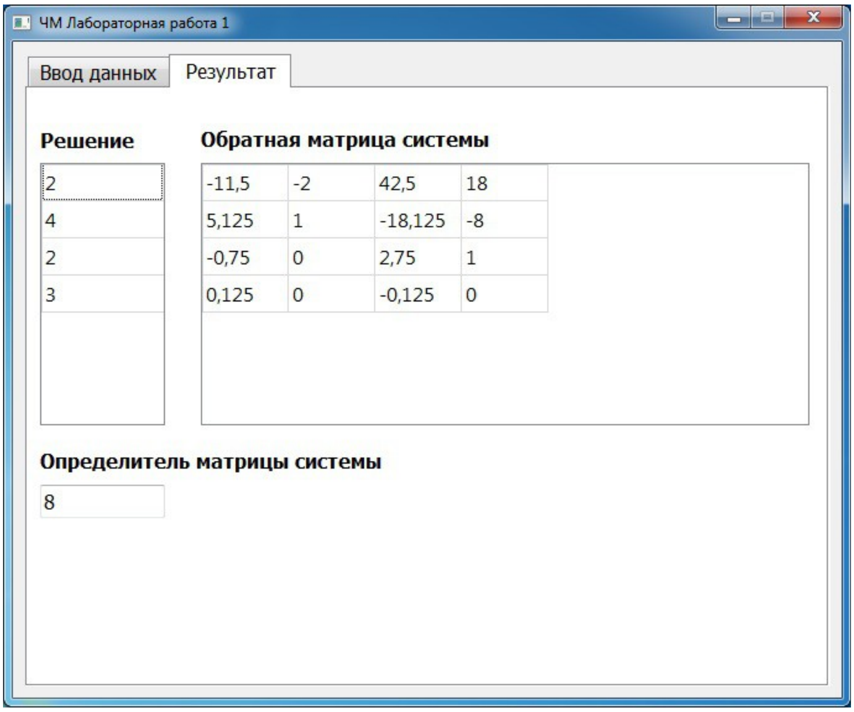
Примечание. Что касается проверок корректности входных данных, выполняются только те из них, без прохождения которых выбранный метод однозначно не может работать (например, попытка решения СЛАУ с не трехдиагональной матрицей методом прогонки). Другие же проверки (например, наличие диагонального преобладания) не производятся, поскольку при их нарушении получение правильного результата возможно. Но следует учитывать, что в таких случаях ничего не гарантируется.

Реализованный метод	Данные, сохраняемые в лог-файл
LUP-разложение	Результат LUP-разложения и проверочная матрица
Метод прогонки	Прогоночные коэффициенты и результат решения СЛАУ
Метод простых итераций	Текущий результат и погрешность для каждой итерации
Метод Зейделя	
Метод вращений (метод Якоби)	Текущий угол поворота, матрица и погрешность для каждой итерации
QR-алгоритм	Результат QR-разложения, проверочные матрицы, а также текущая матрица, результат и погрешности для каждой итерации

Пример ввода.



Результат.



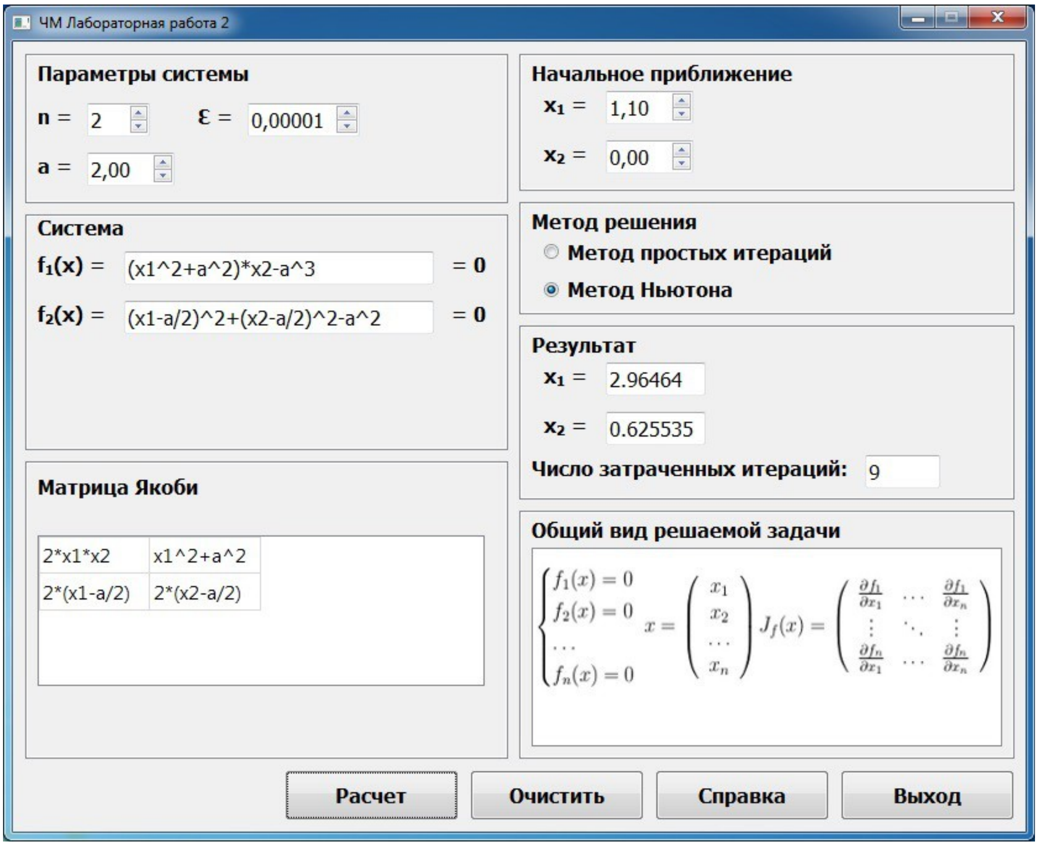
Лабораторная работа 2

Реализованный метод	Решаемая задача
Метод простых итераций	Решение нелинейных уравнений и систем нелинейных уравнений
Метод Ньютона	

Примечание. Стоит обратить внимание, что в данной программе достаточно много нетривиального ввода. К вычислениям, которые необходимо проделать вручную, относится выражение переменных из уравнений (здесь подойдет не любой способ, должно выполняться условие сходимости) и нахождение максимума нормы матрицы Якоби полученных выражений (см. достаточное условие сходимости) в методе простых итераций, а также заполнение матрицы Якоби решаемой системы в методе Ньютона.

Реализованный метод	Данные, сохраняемые в лог-файл
Метод простых итераций	Текущий результат и погрешность для каждой итерации
Метод Ньютона	Текущий результат, погрешность и данные из LUP-разложения для каждой итерации

Пример ввода и соответствующий результат.

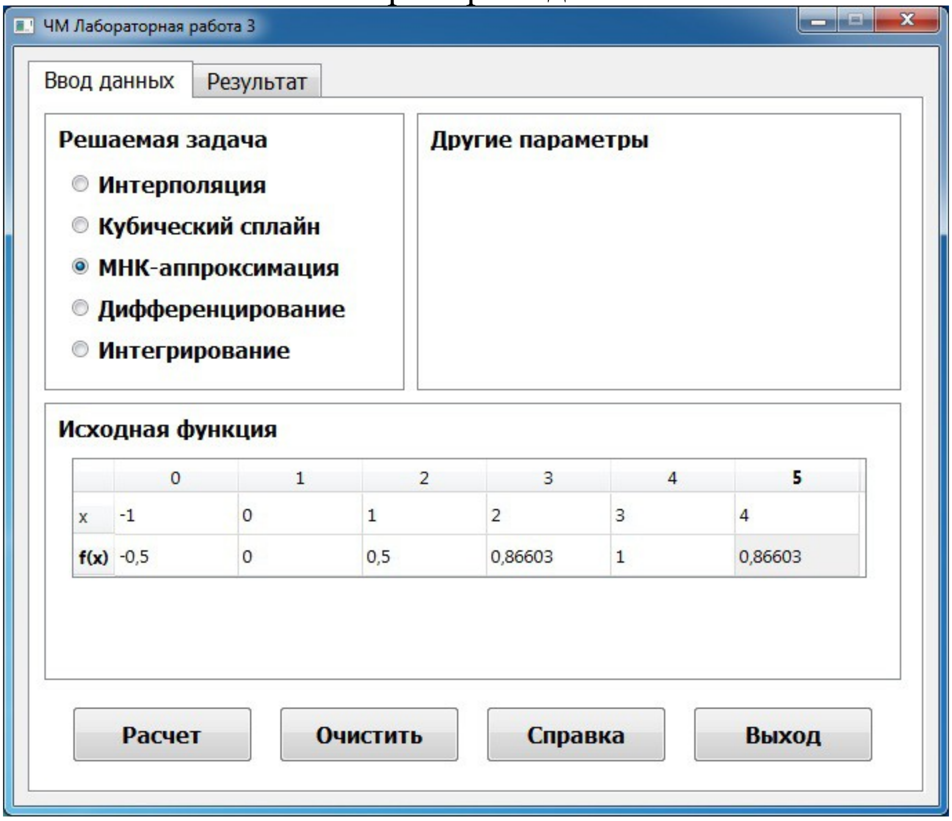


Лабораторная работа 3

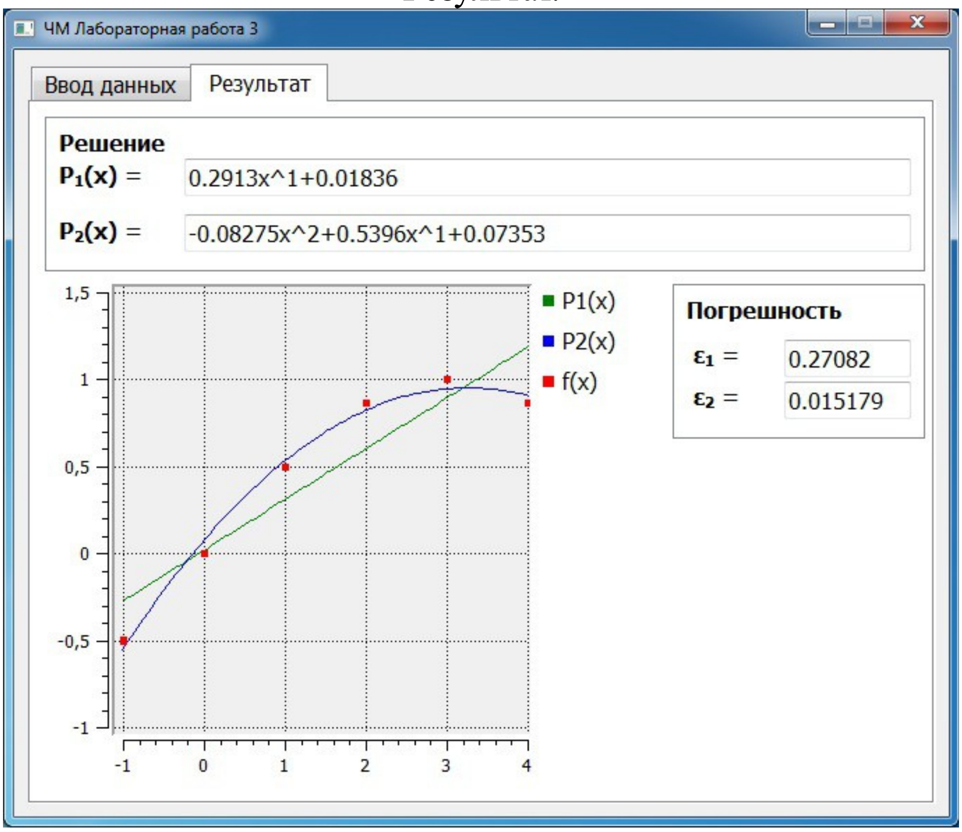
Реализованный метод	Решаемая задача
Построение интерполяционных многочленов Лагранжа и Ньютона	Полиномиальная интерполяция
Построение кубического сплайна	Сплайн-интерполяция
Метод наименьших квадратов (построение приближающих многочленов 1 и 2 степени)	Аппроксимация
Построение интерполяционного многочлена Ньютона и вычисление его 1 и 2 производной	Численное дифференцирование
Метод прямоугольников	Численное интегрирование
Метод трапеций	
Метод Симпсона	

Реализованный метод	Данные, сохраняемые в лог-файл
Построение интерполяционных многочленов Лагранжа и Ньютона	Конечные разности (для многочлена Ньютона)
Построение кубического сплайна	Разности абсцисс между соседними точками (h), матрица и правая часть составленной СЛАУ, данные из метода прогонки, коэффициенты сплайна (a, b, c, d)
Метод наименьших квадратов (построение приближающих многочленов 1 и 2 степени)	Матрица и правая часть составленной СЛАУ, данные из LUP-разложения
Построение интерполяционного многочлена Ньютона и его дифференцирование	Конечные разности (для многочлена Ньютона)
Метод прямоугольников	Границы области, шаг, результат на каждом участке разбиения, общий результат
Метод трапеций	
Метод Симпсона	

Пример ввода.



Результат.

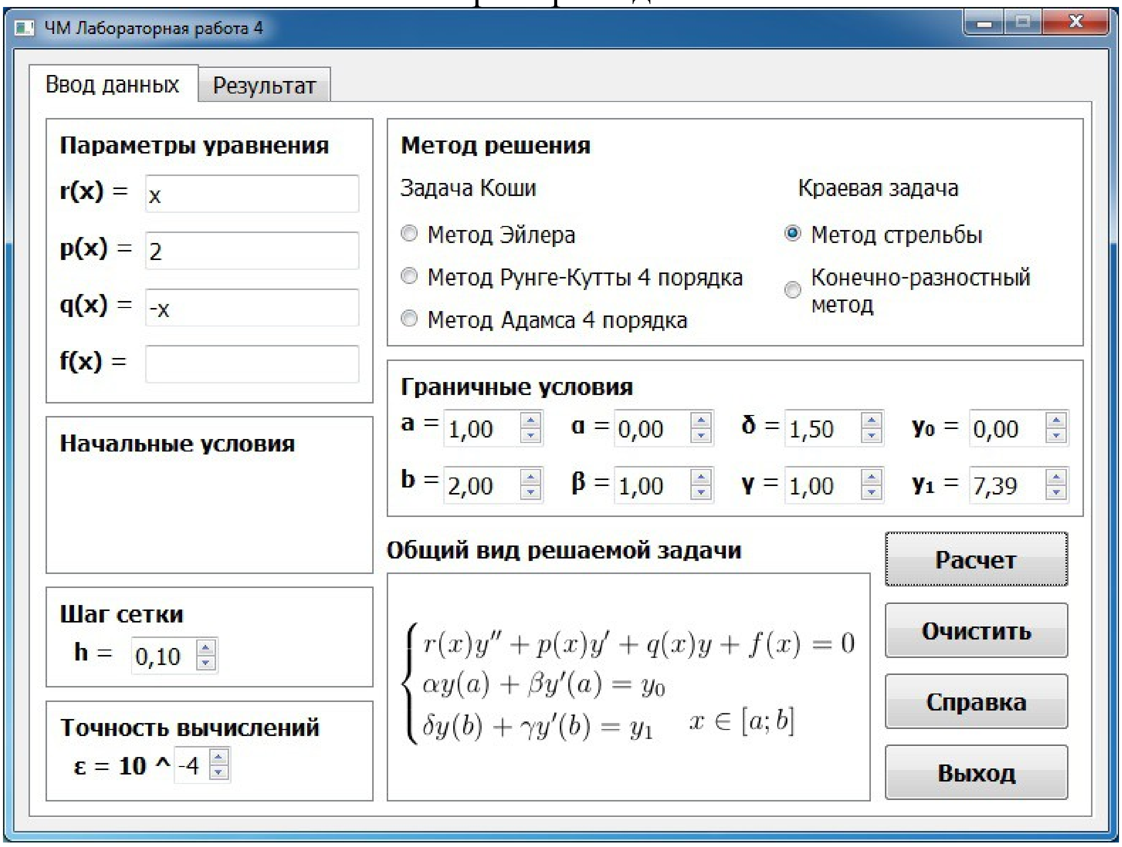


Лабораторная работа 4

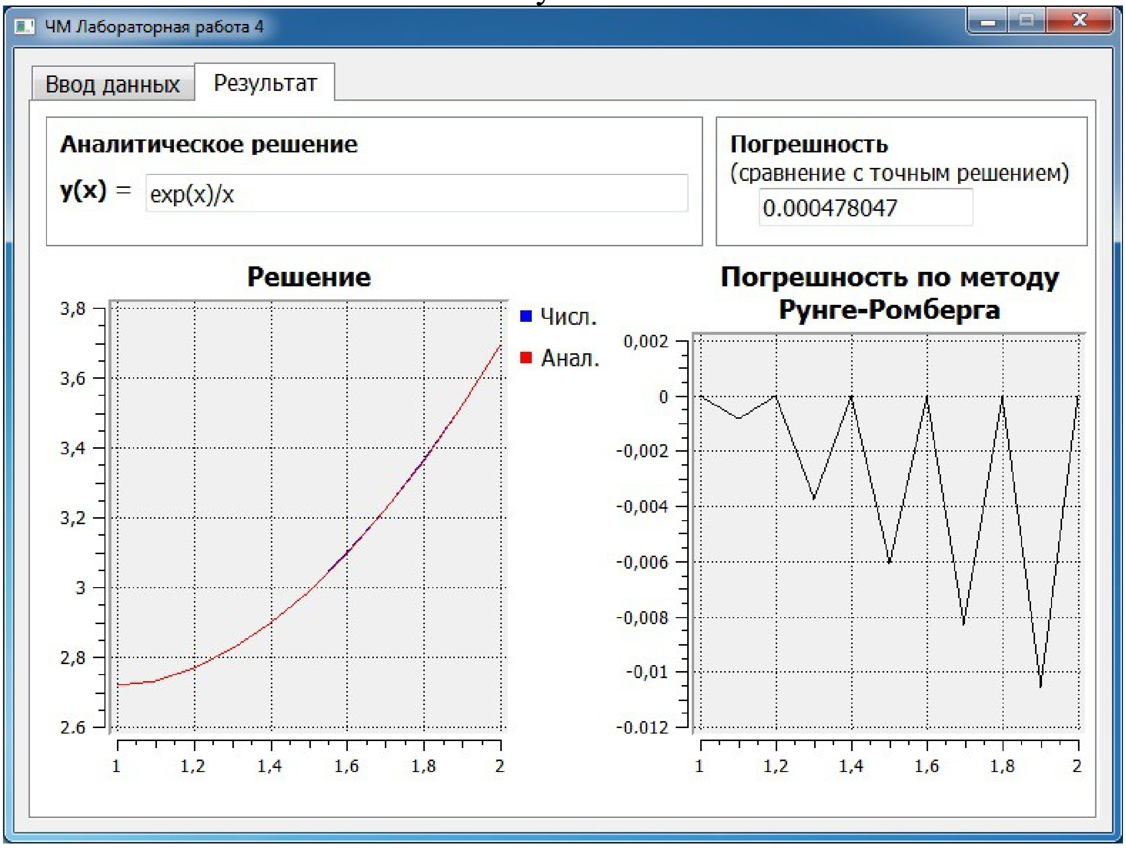
Реализованный метод	Решаемая задача
Метод Эйлера	Задача Коши для ОДУ 2-го порядка
Метод Рунге-Кутты 4-го порядка	
Метод Адамса 4-го порядка	
Метод стрельбы	Краевая задача для ОДУ 2-го порядка
Конечно-разностный метод	

Выходной файл состоит из одной строки, в которой через пробел записаны значения численного решения в узлах сетки (в порядке возрастания x).

Пример ввода.



Результат.



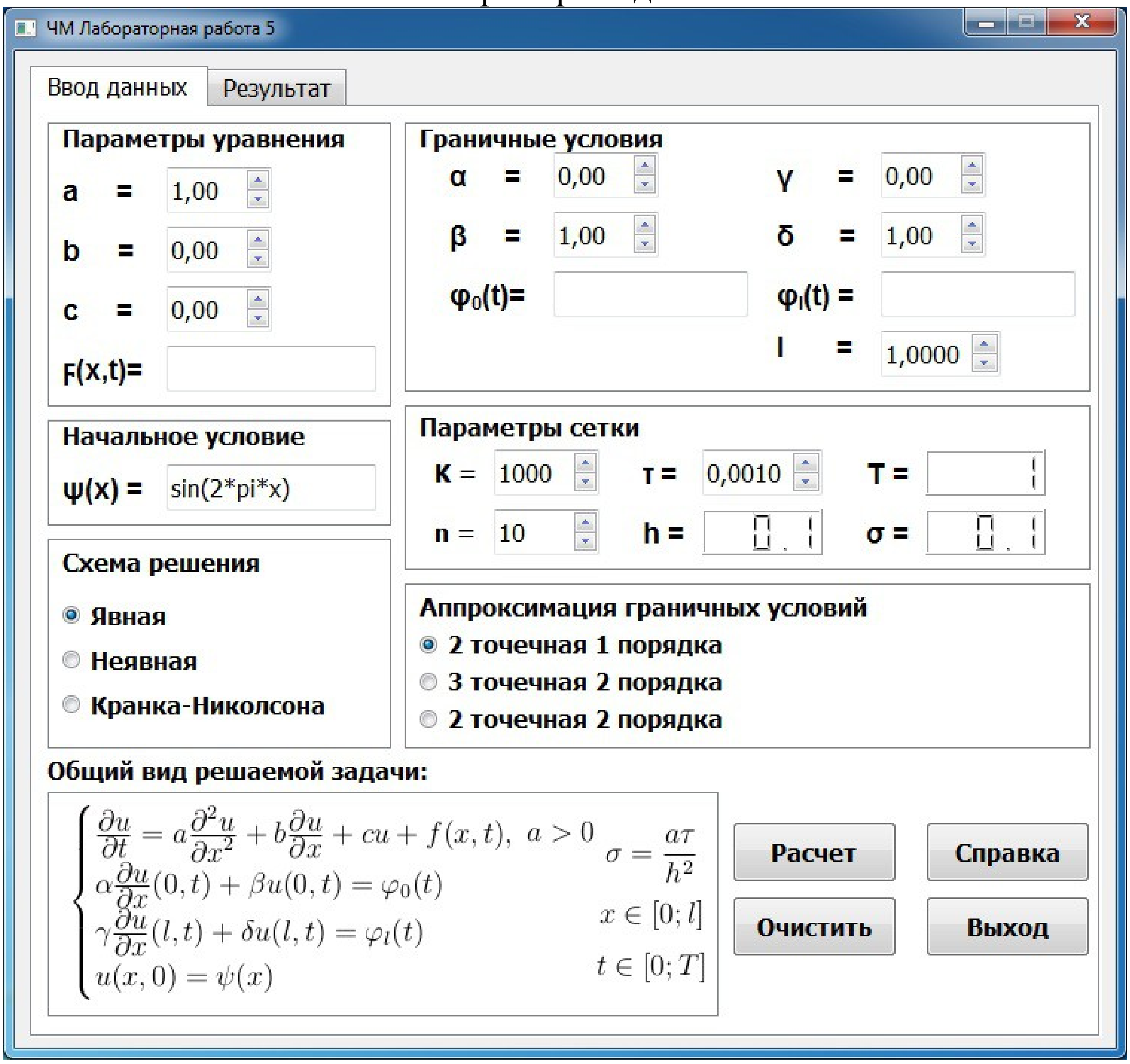
Примечание. Поле «Погрешность (сравнение с точным решением)» вычисляется как максимум модуля разности численного и аналитического решения по всем узлам сетки.

Лабораторная работа 5

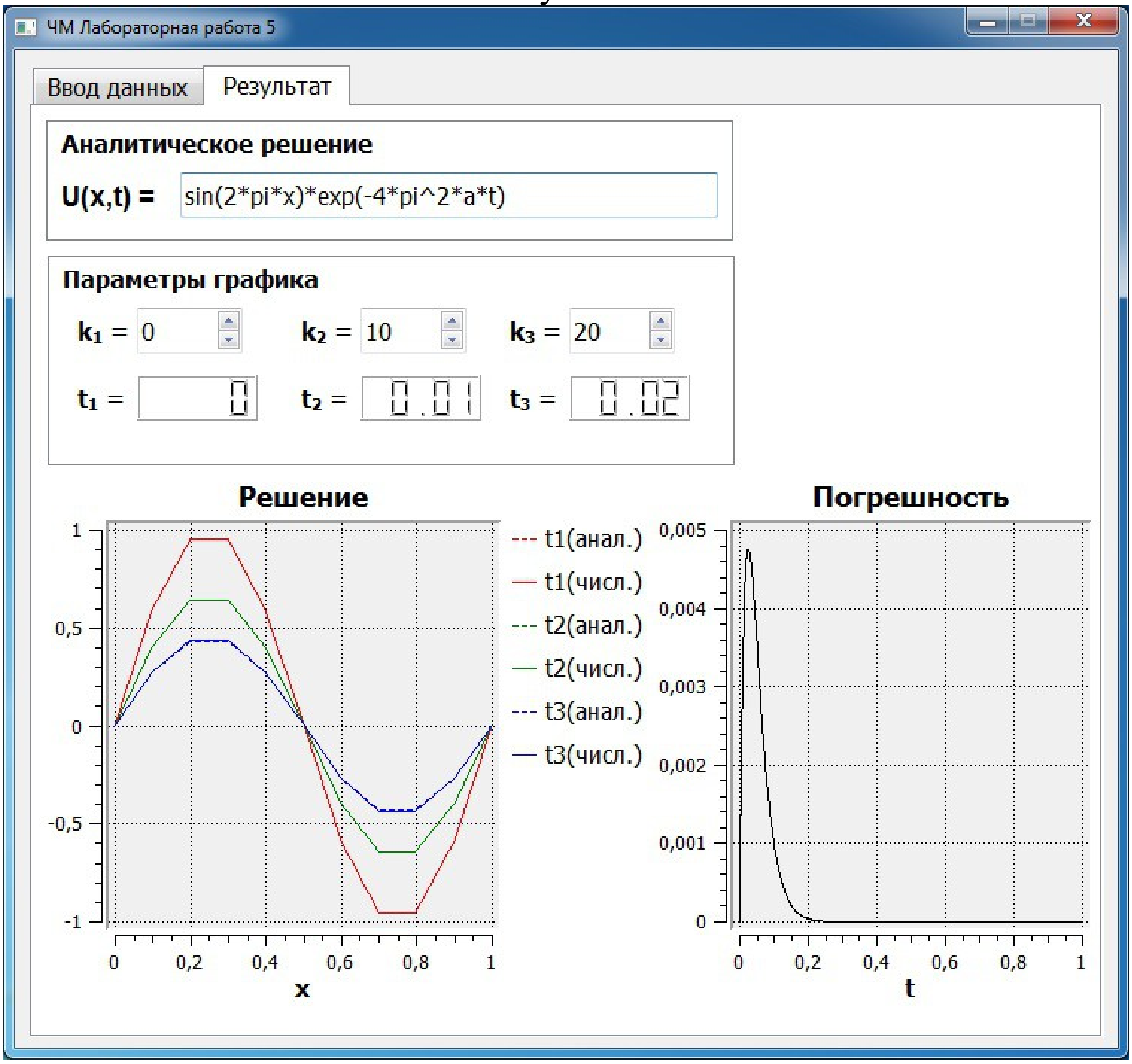
Реализованный метод	Решаемая задача
Явная конечно-разностная схема	Начально-краевая задача для дифференциального уравнения параболического типа
Неявная конечно-разностная схема	
Разностная схема Кранка-Николсона	

Выходной файл состоит из набора строк. Каждая строка соответствует конкретному значению времени (строки расположены в порядке возрастания t). В каждой строке через пробел записаны значения численного решения в узлах сетки (в порядке возрастания x).

Пример ввода.



Результат.



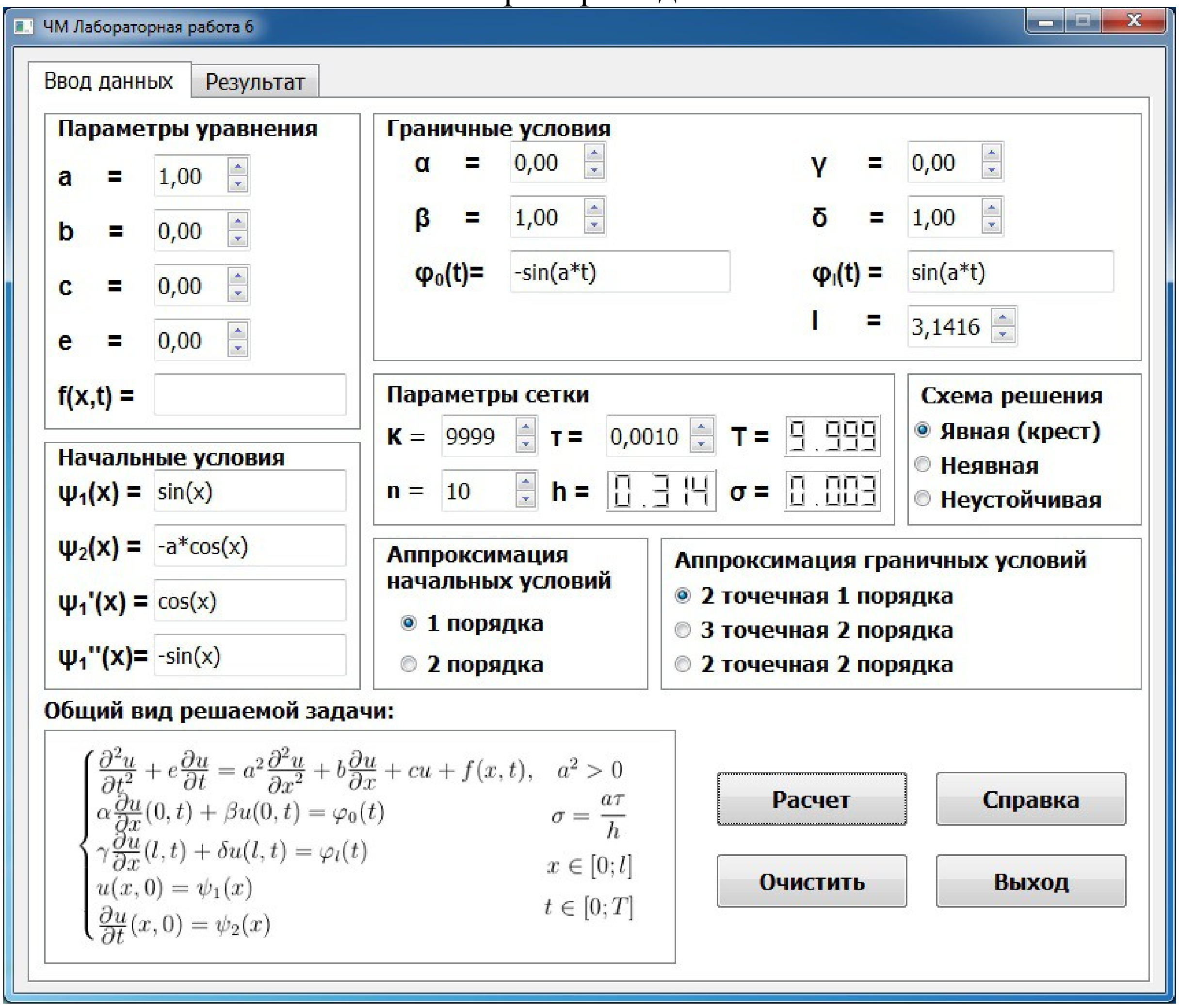
Примечание. Погрешность вычисляется как максимум модуля разности численного и аналитического решения по x при фиксированном t.

Лабораторная работа 6

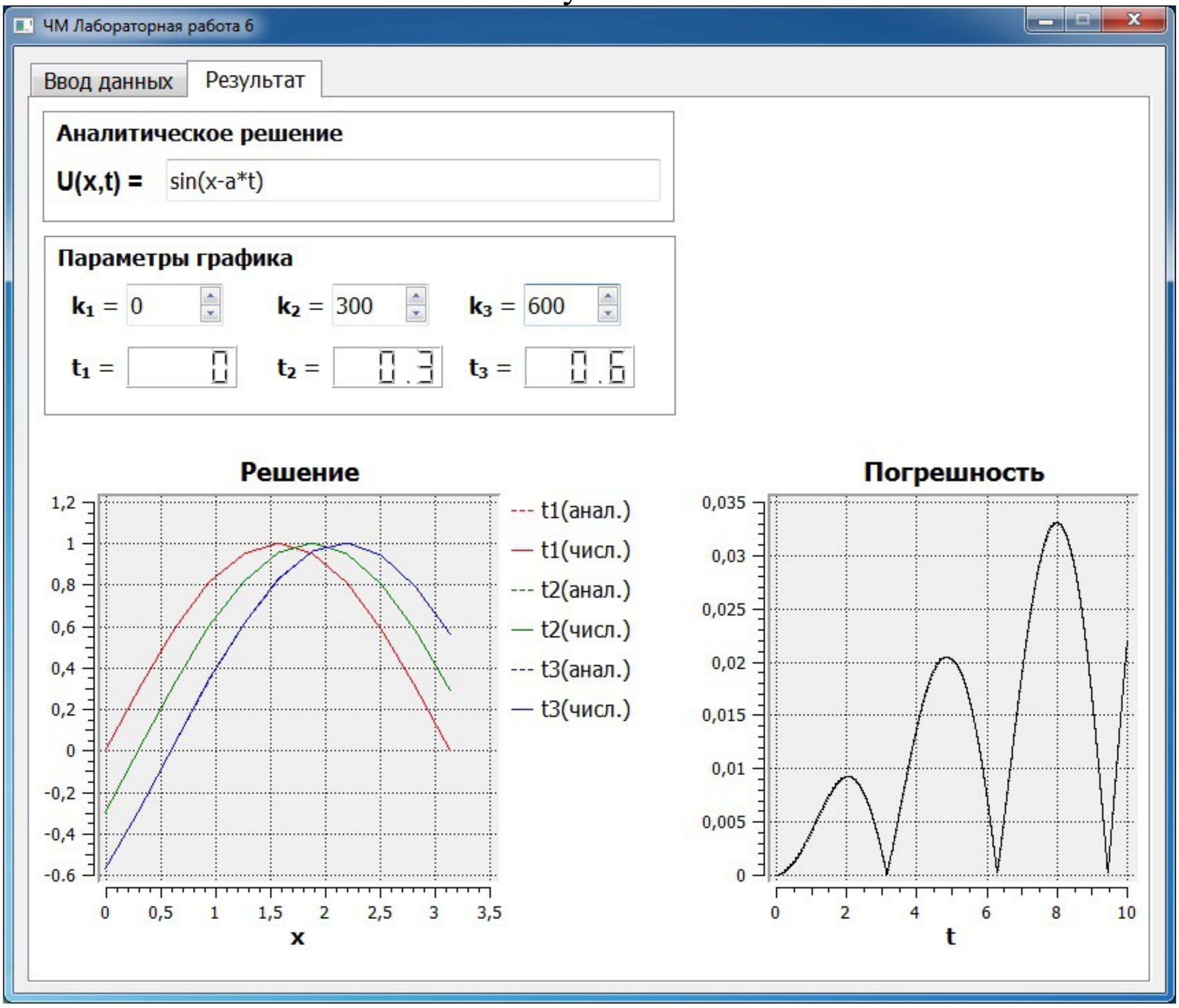
Реализованный метод	Решаемая задача
Явная конечно-разностная схема («крест»)	Начально-краевая задача для дифференциального уравнения гиперболического типа
Неявная конечно-разностная схема	
Неустойчивая конечно-разностная схема	

Выходной файл состоит из набора строк. Каждая строка соответствует конкретному значению времени (строки расположены в порядке возрастания t). В каждой строке через пробел записаны значения численного решения в узлах сетки (в порядке возрастания x).

Пример ввода.



Результат.



Примечание. Погрешность вычисляется как максимум модуля разности численного и аналитического решения по x при фиксированном t.

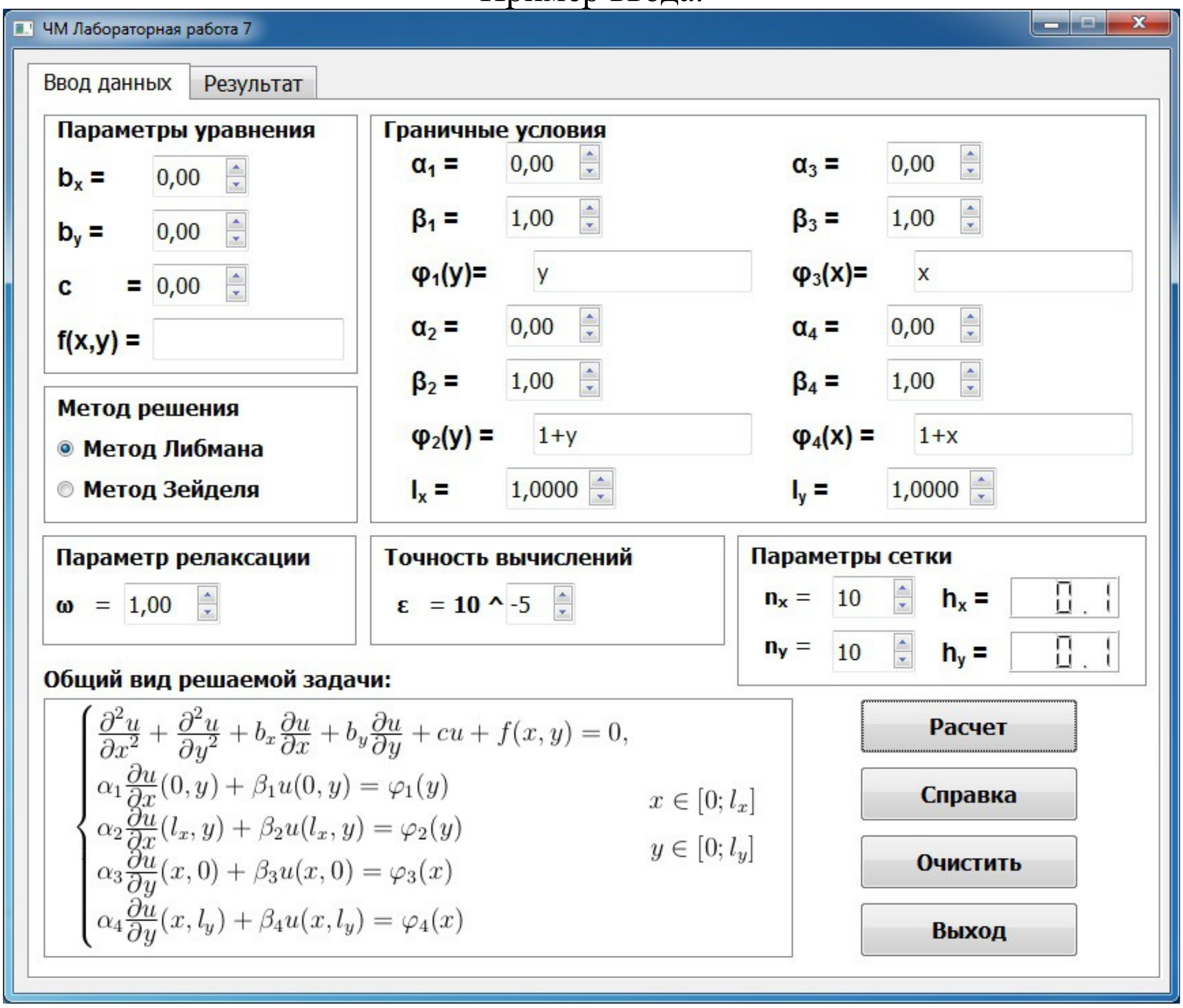
Лабораторная работа 7

Реализованный метод	Решаемая задача
Метод простых итераций (метод Либмана)	Краевая задача для дифференциального уравнения эллиптического типа
Метод Зейделя	

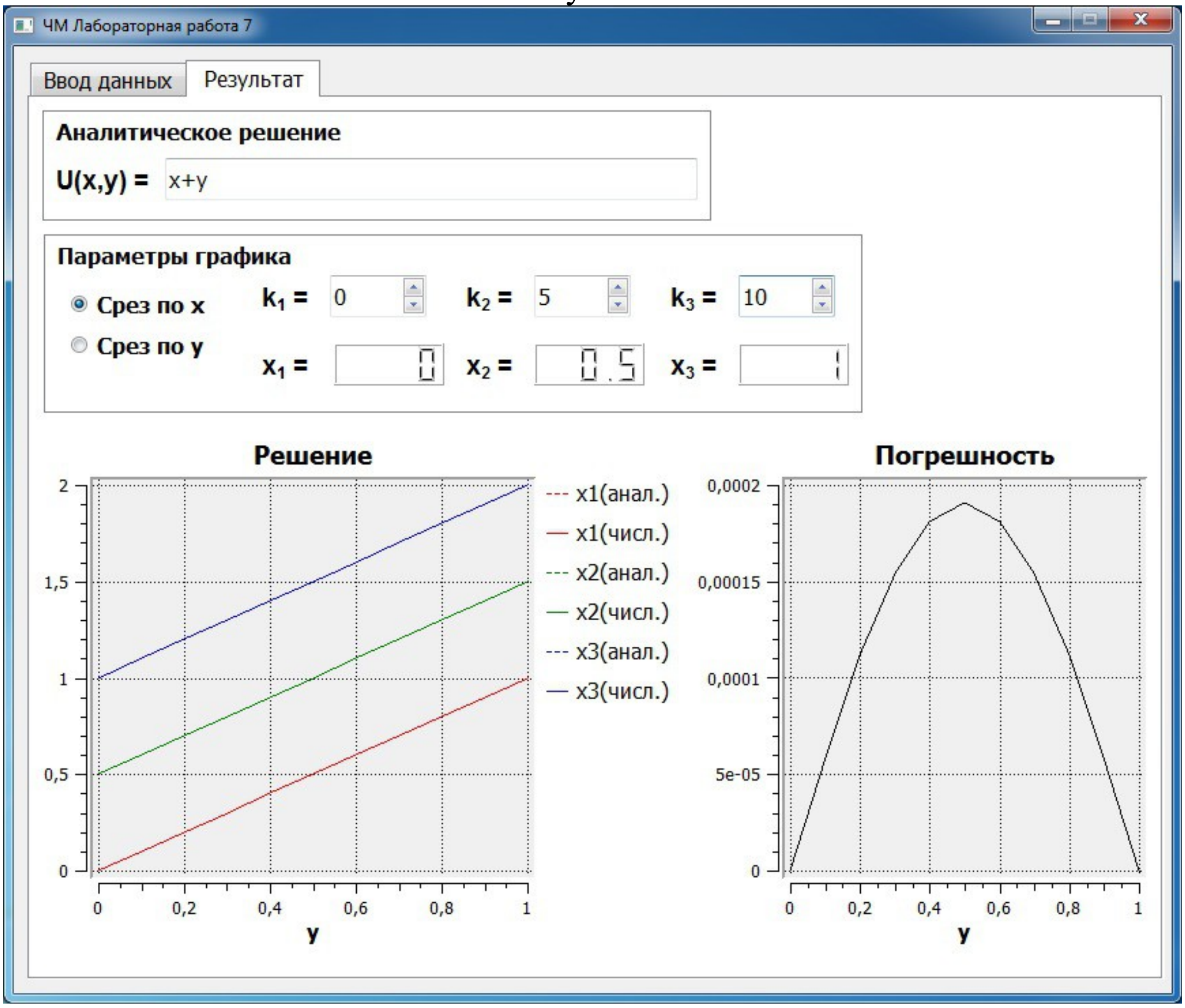
Примечание. В качестве аппроксимации граничных условий используется трехточечная второго порядка. Также вместе с реализованными методами применяется релаксация, коэффициент которой задается во входных данных.

Выходной файл состоит из набора строк. Каждая строка соответствует конкретному значению координаты x (строки расположены в порядке возрастания x). В каждой строке через пробел записаны значения численного решения в узлах сетки (в порядке возрастания y).

Пример ввода.



Результат.



Примечание. Погрешность вычисляется как максимум модуля разности численного и аналитического решения по координате среза при фиксированной другой координате.

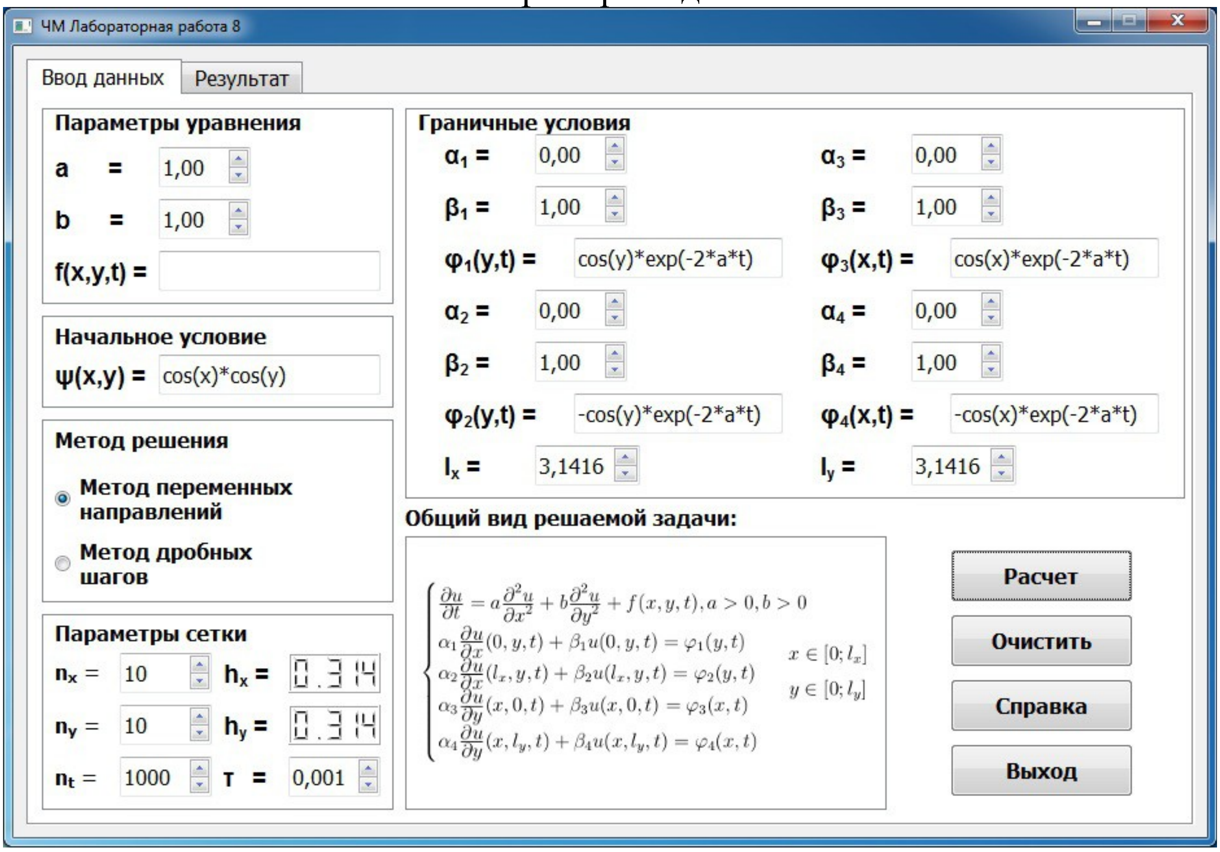
Лабораторная работа 8

Реализованный метод	Решаемая задача
Метод переменных направлений	Двумерная начально-краевая задача для дифференциального уравнения параболического типа
Метод дробных шагов	

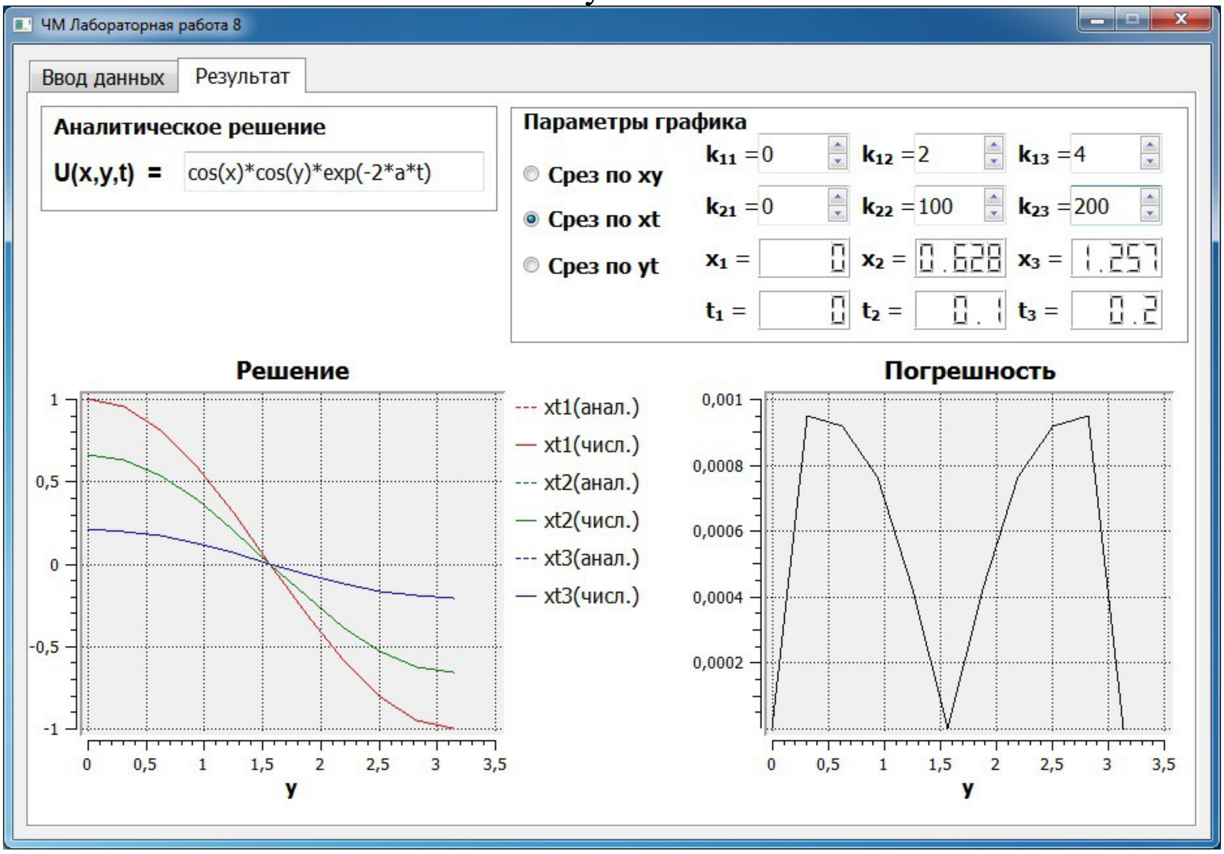
Примечание. В качестве аппроксимации граничных условий используется двухточечная первого порядка.

Выходной файл состоит из набора блоков. Каждый блок соответствует конкретному значению времени (блоки расположены в порядке возрастания t) и состоит из набора строк. Каждая строка в блоке соответствует конкретному значению координаты x (строки в блоке расположены в порядке возрастания x). В каждой строке через пробел записаны значения численного решения в узлах сетки (в порядке возрастания y).

Пример ввода.



Результат.



Примечание. Погрешность вычисляется как максимум модуля разности численного и аналитического решения по координатам среза при фиксированной оставшейся координате.

Заключение

Подводя итоги проделанной работы, стоит отметить, что она была достаточно интересной. Я изучил многие классические численные методы, позволяющие решать известные и широко распространенные задачи из разных областей математики, а также всевозможные особенности, достоинства и недостатки данных методов. Что касается программирования, мне довелось получше узнать фреймворк Qt, а также познакомиться с такими инструментами, как библиотека построения графиков Qwt, модуль юнит-тестирования CppUnit и статический анализатор кода Cppcheck. Кроме того, нужно сказать, что создание графических интерфейсов зачастую является несложной, но весьма трудоемкой задачей, которая требует определенной внимательности и имеет свои «подводные камни». Ну и, конечно же, я надеюсь, что мне удалось не только расширить свои знания, но и создать более-менее полезный набор программ, который может рассматриваться как удачный пример выполнения лабораторных, а также как средство для демонстрации работы реализованных в нем численных методов.