МГТУ ГА

Кафедра вычислительной математики и программирования

**Пояснительная записка** к курсовому проекту

на тему:

**«Решение систем дифференциальных уравнений при**

**помощи неявной схемы Адамса 3-го порядка»**

МОСКВА 2008

**Содержание**

Введение

1. Постановка задачи
2. Описание математических методов решения
3. Описание используемого метода
4. Описание блок-схемы
5. Описание программы
6. Анализ результатов

Заключение

Литература

Приложения

Приложение 1

Приложение 2

Приложение 3

**Введение**

Бурное развитие в последнее десятилетие информационных технологий и компьютерной техники способствует возникновению всё более сложных математических задач, для решения которых без применения численных методов требуется значительное время. Очень часто перед специалистом возникают задачи, не требующие абсолютно точного решения; как правило, требуется найти приближенное решение с заданной погрешностью. Наряду с совершенствованием компьютерной техники происходит процесс совершенствования и численных методов программирования, позволяющих за минимальный отрезок времени получить решение поставленной задачи с заданной степенью точности.

Одной из таких задач является решение систем дифференциальных уравнений. Обыкновенными дифференциальными уравнениями можно описать поведение материальных точек в силовом поле, законы химической кинетики, уравнения электрических цепей и т. д. Ряд физических задач может быть сведён к решению дифференциальных уравнений или системы дифференциальных уравнений. Задача решения системы дифференциальных уравнений имеет важное прикладное значение при решении научных и технических проблем. Кроме того, она является вспомогательной задачей при реализации многих алгоритмов вычислительной математики, математической физики, обработки результатов экспериментальных исследований. Поэтому для инженеров крайне важно грамотно находить решение этой задачи.

**1. Постановка задачи**

Необходимо решить с заданной степенью точности задачу Коши для системы дифференциальных уравнений на заданном интервале [a,b]. Добиться погрешности на втором конце не более 0,0001. Результат получить в виде таблицы значений приближенного и точного решений в точках заданного интервала. Построить графики полученных решений и сравнить их с точным решением.

**Исходные данные:**

– система дифференциальных уравнений вида:



– интервал, на котором ищется решение: [a,b]

– погрешность, с которой ищется решение: е

– формулировка задачи Коши в начальной точке заданного интервала: u(a)=u, v(a)=v



– количество узлов сетки, для которой формируется таблица значений приближенного и точного решений системы: nx

– шаг вывода на экран значений искомых функций в узлах заданной сетки: np

**Выходные данные:**

– таблица значений приближенного и точного решений в узлах заданной сетки;

– графики полученных и точных решений.

**2. Описание математических методов решения задачи**

Конкретная прикладная задача может привести к дифференциальному уравнению любого порядка или к системе таких уравнений. Произвольную систему дифференциальных уравнений любого порядка можно привести к некоторой эквивалентной системе дифференциальных уравнений первого порядка. Среди таких систем выделяют класс систем, разрешённых относительно производной неизвестных функций:

(2.1)



Дифференциальное уравнение или система дифференциальных уравнений имеет бесконечное множество решений. Единственные решения выделяют с помощью дополнительных условий, которым должны удовлетворять искомые решения. В зависимости от вида таких условий рассматривают три типа задач, для которых доказано существование и единственность решений.

Первый тип – это задачи Коши, или задачи с начальными условиями. Для таких задач кроме исходного уравнения в некоторой точке a должны быть заданы начальные условия, т.е. значения функции u1(a),…, um(a):

u1(a)=,…, um(a)= (2.2)



Ко второму типу задач относятся так называемые граничные, или краевые задачи, в которых дополнительные условия задаются в виде функциональных соотношений между искомыми решениями. Количество условий должно совпадать с порядком n уравнения или системы. Если решение задачи определяется в интервале x∈[a,b], то такие условия могут быть заданы как на границах, так и внутри интервала.

Третий тип задач для систем дифференциальных уравнений – это задачи на собственные значения. Такие задачи отличаются тем, что кроме искомых функций u1(x),…, um(x) в уравнения входят дополнительно n неизвестных параметров λ1, λ2, ..., λn, которые называются собственными значениями. Для единственности решения на интервале [a,b] необходимо задать n + m граничных условий.

Рассмотрим подробнее задачу Коши. Воспользуемся компактной записью задачи (2.1), (2.2) в векторной форме:

(2.3)



Требуется найти на интервале [a,b].



Задачу Коши удобнее всего решать методом сеток. Метод сеток состоит в следующем :

1) Выбираем в области интегрирования упорядоченную систему точек a=x1<x2<…<xn<b, называемую сеткой. Точки xi называют узлами разностной сетки, разность между соседними узлами h=xi-xi-1 – шаг сетки. Формула для вычисления шага равномерной сетки, заданной на интервале [a,b]:

, (2.4)



где nx – количество узлов заданной сетки.

2) Решение ищется в виде таблицы значений в узлах выбранной сетки, для чего дифференцирование заменяется системой алгебраических уравнений, связывающих между собой значения искомой функции в соседних узлах. Такую систему уравнений принято называть конечно-разностной схемой.



Для получения конечно-разностной схемы удобно использовать интегроинтерполяционный метод, согласно которому необходимо проинтегрировать уравнение (2.3) на каждом интервале [xk, xk+1] и разделить полученное выражение на длину этого интервала:

(2.5)



Далее апроксимируем интеграл в правой части одной из квадратурных формул и получаем систему уравнений относительно приближенных неизвестных значений искомых функций, которые в отличие от точных обозначим . При этом возникает погрешность ε, обусловленная неточностью апроксимации:



ε(h)=|| || (2.6)



Согласно основной теореме теории метода сеток (теорема Лакса), для устойчивой конечно-разностной схемы при стремлении шага h к нулю погрешность решения стремится к нулю с тем же порядком, что и погрешность апроксимации:

, (2.7)



где С0 – константа устойчивости, p – порядок апроксимации.

Поэтому для увеличения точности решения необходимо уменьшить шаг сетки h.

На практике применяется множество видов конечно-разностных схем, которые подразделяются на одношаговые, многошаговые схемы и схемы с дробным шагом.

**Одношаговые схемы**

**– Метод Эйлера**

Заменяем интеграл в правой части уравнения (2.5) по формуле левых прямоугольников:

(2.8)



Получим:

, (2.9)



где k=0,1,2,…,n.

Схема явная устойчивая. В силу того, что формула для левых прямоугольников имеет погрешность второго порядка, точность ε(h) первого порядка.



– **Неявная схема 1-го порядка**

Используя формулу правых прямоугольников, получим:

(2.10)



Эта схема неразрешима в явном виде относительно , поэтому проводится итерационная процедура:



, (2.11)



где s=1,2,… - номер итерации. Обычно схема сходится очень быстро – 2-3 итерации. Неявная схема первого порядка эффективнее явной, так как константа устойчивости С0 у неё значительно меньше.

– **Метод Эйлера-Коши**

Вычисления проводятся в два этапа : этап прогноза и этап коррекции.

На этапе прогноза определяется приближенное решение на правом конце интервала по методу Эйлера:

(2.12)



На этапе коррекции, используя формулу трапеций, уточняем значение решения на правом конце:

(2.13)



Так как формула трапеций имеет третий порядок точности, то порядок погрешности апроксимации – равен двум.

– **Неявная схема 2-го порядка (метод Эйлера-Коши)**

Используя в (2.5) формулу трапеций, получим:

(2.14)



Схема не разрешена в явном виде, поэтому требуется итерационная процедура:

, (2.15)



где s=1,2,… – номер итерации. Обычно схема сходится за 3-4 итерации.

Так как формула трапеций имеет третий порядок точности, то погрешность апроксимации – второй.

**Схемы с дробным шагом**

– **Схема предиктор-корректор (Рунге-Кутта) 2-го порядка**

Используя в (2.5) формулу средних, получим:

,(2.16)



где – решение системы на середине интервала [xk, xk+1] . Уравнение явно разрешено относительно , однако в правой части присутствует неизвестное значение . Поэтому сначала расчитывают (предиктор):



. (2.17)



Затем расчитывают (корректор) по формуле (2.16). Схема имеет первый порядок погрешности.



**– Схема Рунге-Кутта 4-го порядка**

Используя в (2.5) формулу Симпсона, получим:

(2.18)



Наиболее часто рассчитывают неявное по уравнение по следующей схеме:



Сначала рассчитывают предиктор вида:

(2.19)



затем корректор по формуле:

(2.20)



Поскольку формула Симпсона имеет пятый порядок погрешности, то точность ε(h) – четвёртого порядка.

**Многошаговые схемы**

Многошаговые методы решения задачи Коши характеризуются тем, что решение в текущем узле зависит от данных не в одном предыдущем или последующем узле сетки, как это имеет место в одношаговых методах, а зависит от данных в нескольких соседних узлах.

Идея методов Адамса заключается в том, чтобы для повышения точности использовать вычисленные уже на предыдущих шагах значения



Если заменим в (2.5) подинтегральное выражение интерполяционным многочленом Ньютона, построенного по узлам , то после интегрирования на интервале получим явную экстраполяционную схему Адамса. Если заменим в (2.5) подинтегральное выражение на многочлен Ньютона, построенного по узлам , то получим неявную интерполяционную схему Адамса.



**– Явная экстраполяционная схема Адамса 2-го порядка**

(2.21)



Схема двухшаговая, поэтому необходимо для расчётов найти по схеме Рунге-Кутта 2-го порядка , после чего , , … вычисляют по формуле (2.21)



**– Явная экстраполяционная схема Адамса 3-го порядка**

(2.22)



Схема двухшаговая, поэтому необходимо сперва найти и по схеме предиктор-корректор 4-го порядка, после чего , , … вычисляют по формуле (2.22).



**3. Описание используемого метода**

Для решения системы дифференциальных уравнений выбрана неявная схема Адамса 3-го порядка, как одна из наиболее точных конечноразностных схем для решения задачи Коши. Чтобы прийти к неявной схеме Адамса, заменим подинтегральное выражение в уравнении:

(3.1)



интерполяционным многочленом Ньютона 2-го порядка, вида:

(3.2)



После интегрирования полученного выражения на интервале , приходим к уравнению неявной схемы Адамса 3-го порядка:



. (3.3)



Данная схема не разрешена явно относительно , поэтому сначала необходимо вычислить любым подходящим методом, например методом Рунге-Кутта четвёртого порядка. Затем для нахождения требуется использовать метод простой итерации:



, (3.4)



где s=1,2,3,… – номер итерации. Условие выхода из цикла итерационной процедуры:

, (3.5)



где ε – заданная погрешность.

Начальное приближение задаётся формулой для явной экстраполяционной схемы Адамса 2-го порядка:

. (3.6)



Схема устойчива, сходится быстро. Чаще всего достаточно одной итерации. Порядок погрешности ε(h) неявной схемы Адамса третьего порядка равен четырём.

**4. Описание блок-схемы алгоритма**

При разработке программы были построены блок-схемы алгоритма программы, упрощающие процесс проектирования и облегчающие понимание исходного кода готовой программы (см. Приложение 1).

Блок-схема алгоритма условно разбита на 11 блоков.

Главная функция программы (блоки 1,2,5) отвечает за обработку события создания формы, взаимодействие со стандартным компонентом TСhart, а также за реализацию решения системы дифференциальных уравнений неявной схемой Адамса 3-го порядка. Блок-схема алгоритма решения задачи Коши разбита условно на 35 блоков:

1-й блок отвечает за ручной ввод интервала [a,b], на котором ищется решение системы; количества шагов сетки nx; шаг вывода результатов на экран np; строк u1 и v1, соответствующих уравнениям системы; значения искомых функций в начале заданного интервала; допустимая погрешность e.

Во втором блоке происходит вычисление шага h и установка текущего узла на x=a. Блок 3 – функция преобразования исходных записей уравнений системы в равносильные им строки с постфиксной формой записью математических операций (см. далее «алгоритм обратной польской записи»). В качестве аргументов функции выступают введённые ранее строки u1 и v1.

Блоки 4-15 – расчет первых 2-х точек заданной сетки методом Рунге-Кутта 4-го порядка. В данных блоках и далее используется пользовательская функция FPR, рассчитывающая значения вводимых пользователем уравнений в узлах заданной сетки. В качестве аргументов функции выступают: уже преобразованные в обратную польскую запись строки, задающие уравнения системы; текущее значение x; значения искомых функций на предыдущем шаге (условно обозначаем ).



В блоках 16-34 в цикле (16) рассчитываются значения искомых решений в узлах 2-nx заданной сетки неявной схемой Адамса 3-го порядка. Цикл 21-29 осуществляет итерационную процедуру неявной схемы. Условие выхода из этого цикла – выполнение неравенства de<e, где de – наибольший из модулей , e – заданная точность. Поскольку на экран выводятся значения искомых функций не во всех узлах, а только в узлах с номером, кратным шагу вывода nx, вводимым с клавиатуры, то блоки 33-34 осуществляют выбор этих узлов.



Преобразование в обратную польскую запись происходит по следующим правилам:

Рассматриваем поочередно каждый символ:

1. Если этот символ - число (или переменная), то просто помещаем его в выходную строку.

2. Если символ - знак операции (+, -, \*, / ,^), то проверяем приоритет данной операции. Операция возведения в степень имеет наивысший приоритет (равен 4). Операции сложения и вычитания имеют меньший приоритет (равен 2). Наименьший приоритет (равен 1) имеет открывающая скобка.

Получив один из этих символов, мы должны проверить стек:

а) Если стек все еще пуст, или находящиеся в нем символы (а находится в нем могут только знаки операций и открывающая скобка) имеют меньший приоритет, чем приоритет текущего символа, то помещаем текущий символ в стек.

б) Если символ, находящийся на вершине стека имеет приоритет, больший или равный приоритету текущего символа, то извлекаем символы из стека в выходную строку до тех пор, пока выполняется это условие; затем переходим к пункту а).

3. Если текущий символ - открывающая скобка, то помещаем ее в стек.

4. Если текущий символ - закрывающая скобка, то извлекаем символы из стека в выходную строку до тех пор, пока не встретим в стеке открывающую скобку (т.е. символ с приоритетом, равным 1), которую следует просто уничтожить. Закрывающая скобка также уничтожается.

Если вся входная строка разобрана, а в стеке еще остаются знаки операций, извлекаем их из стека в выходную строку.

Согласно этим правилам создан модуль ”Unit3.cpp”, содержащий функцию преобразования строки в обратную польскую запись OPZ (блок 3 в блок-схеме алгоритма), алгоритм которой приведён в приложении. В данном модуле использованы также вспомогательные функции PUSH, PRIOR, DEL. Функция PUSH записывает в стек, на вершину которого указывает HEAD, символ a. Возвращает указатель на новую вершину стека. Функция PRIOR вычисляет приоритет текущего символа, естественно, лишь в том случае, если текущий символ – математическая операция. Функция DEL удаляет символ с веpшины стека. Возвpащает удаляемый символ. Изменяет указатель на веpшину стека.

Для работы с полученной обратной польской записью создана функция(блок 4), организованная в виде подключаемого модуля “Unit5.cpp”. Блок-схема данной функции приведена в приложении. На начальном этапе (блоки 1-13) в цикле анализируется строка, содержащая обратную польскую запись. Если символ ранее задекларирован (‘x’,’u’,’v’,’e’,’1’..’9’), то его значение заносится в текущий элемент массива th. На следующем этапе (блоки 14-29) осуществляется «обратный ход» польской нотации: анализируется каждый символ строки, и если этот символ ранее задекларирован, то его значение помещается в стек (блоки 15-17). В случае, если текущий символ – знак математической операции , то из стека извлекаются последние два элемента и с ними проводится указанная операция. Результат заносится на вершину стека. Стек в функции реализован в виде однонаправленного массива типа double. Функция возвращает первый элемент стека.

**5. Описание программы**

После проведённого обзора программных средств для разработки данного программного продукта, была выбрана среда Borland C++ Builder. Язык С++ хорошо зарекомендовал себя эффективностью, лаконичностью записи алгоритмов, логической стройностью программы, хорошей переносимостью. Программы, написанные на языке С++, сравнимы по скорости с программами, написанными на языке ассемблера; при этом они более наглядны и просты в сопровождении. Среда Borland C++ Builder является средством быстрой разработки windows-приложений, позволяющее создавать приложения на языке С++, используя среду разработки и библиотеку компонентов Delphi.

Готовая программа представляет собой исполняемый файл с именем “Adams3.exe”, реализованный в виде Widows-приложения в среде Borland C++ Builder. После запуска программы на рабочем окне появляется рабочее окно с заголовком «Решение систем дифференциальных уравнений» ( см. Приложение 3, рис.1). В активном окне можно выделить следующие области:

1. Область ввода исходных данных.
2. Окно вывода результатов.
3. Поле отображения графиков полученных функций, являющихся

решением заданной системы, и графиков истинного решения.

1. Основное меню.

1) Область исходных данных содержит поля, в которые требуется ввести начальные данные: систему дифференциальных уравнений; интервал, на котором требуется найти решение заданной системы; допустимую погрешность; условия Коши в начальной точке заданного интервала; количество шагов “сетки” и шаг вывода полученных значений искомых функций в узлах сетки.

В поля ”du/dx= “ и “dv/dx= “ вводятся дифференциальные уравнения, содержащие символы, ‘u’, ‘v’ ‘x’, ‘e’, ’1’..’9’, ’+’, ’-‘, ‘\*’, ‘/’, ‘^’, ‘(‘, ‘)’. Здесь: символы ‘u’ и ‘v’ представляют собой искомые функции, символ ‘e’ является основанием натурального логарифма, символ ‘^’ обозначает операцию возведения в степень. Использование других символов нежелательно, так как они будут проигнорированы программой.

Поля с заглавием «интервал [a;b]» содержат начальную и конечную точку промежутка, на котором будет найдено решение заданной системы.

В поле «количество шагов сетки» требуется ввести целое число, равное количеству точек по оси OX на заданном интервале, в которых ищем значения функций u(x) и v(x).

Поле «шаг вывода» содержит целое число, определяющее частоту вывода на экран результатов из множества результатов во всех узлах заданной сетки.

Поля под общим названием «начальные условия» содержат условия Коши – значения искомых функций в начале заданного интервала [a,b].

Для корректной работы программы все поля должны быть заполнены. При запуске программы все вышеперечисленные поля уже содержат стандартную информацию для теста программы, которую можно изменять.

Пользователю предоставляется возможность выбора режима программы. При запуске программы метка возле надписи «Не использовать метод сгущающихся сеток» отсутствует, и программа, используя метод учащающихся сеток подберёт после первого нажатия кнопки «выполнит» оптимальное значение количества шагов для достижения заданной точности. После повторного нажатия кнопки «выполнить» будут произведены вычисления уже для рекомендуемого значения шага сетки. Если метка поставлена, то после нажатия кнопки «выполнить» будет решена задача Коши для заданного интервала, но заданная точность не будет достигнута. Данный режим позволяет вводить различные системы дифференциальных уравнений, отличных от стандартных тестовых, решением которых являются функции u(x)=2\*x, v(x)=exp(x).

2) Все результаты, полученные в ходе работы программы, отображаются в отдельном окне (рис. 2). При желании, всю информацию из этого окна можно сохранить в отдельный файл.

3) Полученное решение в виде графиков искомых функций выводится в отдельное поле (рис. 2). Здесь отображаются также графики функций f(x)=2\*x и f(x)=exp(x), являющихся точным решением для тестовых систем дифференциальных уравнений. Поле отображения графиков масштабируемо.

4) Основное меню содержит следующие пункты: «Файл» и «О программе» (рис. 3). В свою очередь пункт меню «Файл» содержит следующие подпункты: «новый», «открыть», «сохранить как…» и «выход».

При выборе пункта «новый» все поля и окна будут очищены. Поле отображения графиков будет также очищено.

Выбрав пункт «сохранить как…», вся информация из окна результатов будет сохранена в выбранный пользователем файл (по умолчанию с расширением .txt).

Выбор пункта «открыть» приводит к загрузке из уже сохранённого ранее файла системы дифференциальных уравнений.

Программа работает стабильно, не приводит к ошибкам.

**6. Анализ результатов**

Результатом работы программы “Adams3.exe” является таблица значений полученного решения в узлах заданной сетки, значений точного решения и разность между точным и полученным решениями. Данную таблицу можно сохранить в текстовый файл с возможностью дальнейшего просмотра и редактирования.

В качестве тестовой задачи была решена задача Коши при помощи неявной схемы Адамса 3-го порядка на интервале [2,4] с начальными условиями :



.



Точным решением данной системы являются функции:



Требовалось добиться решения системы дифференциальных уравнений с точностью до 0.0001.

Результат решения (выходной файл):

Входные данные:

du/dx= u/x+v-e^x;

dv/dx= 2\*x/u+v^2/e^x-1;

Интервал: [2;4]

Допустимая погрешность: е=0,0001

Начальные условия:

u=4

v=7,389056098930650230

Количество шагов сетки: 320

Шаг вывода: 32

Результаты:

x | u(x) | точное | разн. | v(x) | точное | разн. |

2,000 4,0000 4,0000 0,0000 7,3891 7,3891 0,0000

2,200 4,4000 4,4000 0,0000 9,0250 9,0250 0,0000

2,400 4,8000 4,8000 0,0000 11,0232 11,0232 0,0000

2,600 5,2000 5,2000 0,0000 13,4637 13,4637 0,0000

2,800 5,6000 5,6000 0,0000 16,4446 16,4446 0,0000

3,000 6,0000 6,0000 0,0000 20,0855 20,0855 0,0000

3,200 6,4000 6,4000 0,0000 24,5325 24,5325 0,0000

3,400 6,8000 6,8000 0,0000 29,9641 29,9641 0,0000

3,600 7,2000 7,2000 0,0000 36,5982 36,5982 0,0000

3,800 7,6000 7,6000 0,0000 44,7012 44,7012 0,0000

4,000 8,0000 8,0000 0,0000 54,5981 54,5982 0,0000

Время выполнения: 0,015с

Как видно из полученного результата, точность в 0.0001 достигается уже при количестве шагов, равном 320. Время. Затраченное на расчёт таблицы значений на заданном интервале составляет всего 0.015 секунд, что практически не ощутимо. Увеличение шага сетки приведёт к повышению точности решения, однако это увеличит и время работы вычислительного процесса.

Заданная точность достигается за минимальное количество итерраций (1-3 итерации).

Ниже приведен график функций полученного и точного решений:

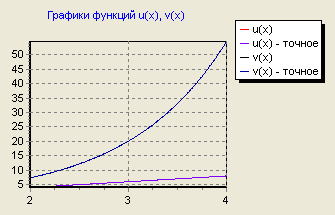


Рис. 5.1 График полученного и точного решения

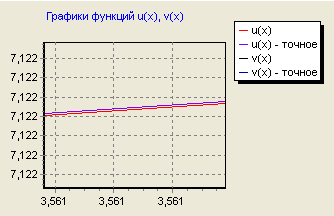


Рис. 5.2 График полученного и точного решения

Как видно из рисунков 5.1, 5.2, расхождение кривых наблюдается только при достаточно большом увеличении графика.

Предложенная задача Коши была также решена в математическом пакете “ Mathcad 11” двумя методами: методом Рунге-Кутта 5-го порядка и методом Рунге-Кутта с непостоянным шагом. Реализация решения системы дифференциальных уравнений в “ Mathcad 11” и таблицы результатов приведены ниже:

Реализация решения задачи Коши методом Рунге-Кутта 5-го порядка:



Таблица 5.1 – Результаты решения задачи Коши методом Рунге-Кутта 5-го порядка.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | u(x) | v(x) |  | x | u(x) | v(x) |
| 2 | 4 | 7,3890561 |  | 3,1 | 6,2 | 22,19795 |
| 2,02 | 4,04 | 7,5383249 |  | 3,12 | 6,24 | 22,64638 |
| 2,04 | 4,08 | 7,6906092 |  | 3,14 | 6,28 | 23,10387 |
| 2,06 | 4,12 | 7,8459698 |  | 3,16 | 6,32 | 23,5706 |
| 2,08 | 4,16 | 8,0044689 |  | 3,18 | 6,36 | 24,04675 |
| 2,1 | 4,2 | 8,1661699 |  | 3,2 | 6,4 | 24,53253 |
| 2,12 | 4,24 | 8,3311375 |  | 3,22 | 6,44 | 25,02812 |
| 2,14 | 4,28 | 8,4994376 |  | 3,24 | 6,48 | 25,53372 |
| 2,16 | 4,32 | 8,6711376 |  | 3,26 | 6,52 | 26,04954 |
| 2,18 | 4,36 | 8,8463062 |  | 3,28 | 6,56 | 26,57577 |
| 2,2 | 4,4 | 9,0250135 |  | 3,3 | 6,6 | 27,11264 |
| 2,22 | 4,44 | 9,2073308 |  | 3,32 | 6,64 | 27,66035 |
| 2,24 | 4,48 | 9,3933313 |  | 3,34 | 6,68 | 28,21913 |
| 2,26 | 4,52 | 9,5830891 |  | 3,36 | 6,72 | 28,78919 |
| 2,28 | 4,56 | 9,7766804 |  | 3,38 | 6,76 | 29,37077 |
| 2,3 | 4,6 | 9,9741824 |  | 3,4 | 6,8 | 29,9641 |
| 2,32 | 4,64 | 10,175674 |  | 3,42 | 6,84 | 30,56941 |
| 2,34 | 4,68 | 10,381237 |  | 3,44 | 6,879999 | 31,18696 |
| 2,36 | 4,72 | 10,590951 |  | 3,46 | 6,919999 | 31,81698 |
| 2,38 | 4,76 | 10,804903 |  | 3,48 | 6,959999 | 32,45972 |
| 2,4 | 4,8 | 11,023176 |  | 3,5 | 6,999999 | 33,11545 |
| 2,42 | 4,84 | 11,245859 |  | 3,52 | 7,039999 | 33,78443 |
| 2,44 | 4,88 | 11,473041 |  | 3,54 | 7,079999 | 34,46692 |
| 2,46 | 4,92 | 11,704811 |  | 3,56 | 7,119999 | 35,1632 |
| 2,48 | 4,96 | 11,941264 |  | 3,58 | 7,159999 | 35,87354 |
| 2,5 | 4,9999999 | 12,182494 |  | 3,6 | 7,199999 | 36,59823 |
| 2,52 | 5,0399999 | 12,428597 |  | 3,62 | 7,239999 | 37,33757 |
| 2,54 | 5,0799999 | 12,679671 |  | 3,64 | 7,279999 | 38,09184 |
| 2,56 | 5,1199999 | 12,935817 |  | 3,66 | 7,319999 | 38,86134 |
| 2,58 | 5,1599999 | 13,197138 |  | 3,68 | 7,359999 | 39,64639 |
| 2,6 | 5,1999999 | 13,463738 |  | 3,7 | 7,399999 | 40,4473 |
| 2,62 | 5,2399999 | 13,735723 |  | 3,72 | 7,439999 | 41,26439 |
| 2,64 | 5,2799999 | 14,013204 |  | 3,74 | 7,479999 | 42,09799 |
| 2,66 | 5,3199999 | 14,296289 |  | 3,76 | 7,519999 | 42,94842 |
| 2,68 | 5,3599999 | 14,585093 |  | 3,78 | 7,559999 | 43,81604 |
| 2,7 | 5,3999999 | 14,879732 |  | 3,8 | 7,599999 | 44,70118 |
| 2,72 | 5,4399999 | 15,180322 |  | 3,82 | 7,639999 | 45,60421 |
| 2,74 | 5,4799999 | 15,486985 |  | 3,84 | 7,679999 | 46,52547 |
| 2,76 | 5,5199999 | 15,799843 |  | 3,86 | 7,719999 | 47,46535 |
| 2,78 | 5,5599999 | 16,119021 |  | 3,88 | 7,759999 | 48,42421 |
| 2,8 | 5,5999999 | 16,444647 |  | 3,9 | 7,799999 | 49,40245 |
| 2,82 | 5,6399999 | 16,776851 |  | 3,92 | 7,839999 | 50,40044 |
| 2,84 | 5,6799999 | 17,115765 |  | 3,94 | 7,879999 | 51,4186 |
| 2,86 | 5,7199999 | 17,461527 |  | 3,96 | 7,919999 | 52,45732 |
| 2,88 | 5,7599999 | 17,814273 |  | 3,98 | 7,959998 | 53,51703 |
| 2,9 | 5,7999998 | 18,174145 |  | 4 | 7,999998 | 54,59815 |
| 2,92 | 5,8399998 | 18,541287 |  |  |  |  |
| 2,94 | 5,8799998 | 18,915846 |  |  |  |  |
| 2,96 | 5,9199998 | 19,297972 |  |  |  |  |
| 2,98 | 5,9599998 | 19,687816 |  |  |  |  |
| 3 | 5,9999998 | 20,085537 |  |  |  |  |
| 3,02 | 6,0399998 | 20,491291 |  |  |  |  |
| 3,04 | 6,0799998 | 20,905243 |  |  |  |  |
| 3,06 | 6,1199998 | 21,327557 |  |  |  |  |
| 3,08 | 6,1599998 | 21,758402 |  |  |  |  |

Реализация решения задачи Коши методом Рунге-Кутта с непостоянным шагом:



Таблица 5.2 – Результаты решения задачи Коши методом Рунге-Кутта с непостоянным шагом.

|  |  |  |
| --- | --- | --- |
| X | u(x) | v(x) |
| 2 | 4 | 7,389056099 |
| 2,2 | 4,4 | 9,025013486 |
| 2,4 | 4,8 | 11,02317634 |
| 2,6 | 5,2 | 13,46373796 |
| 2,8 | 5,6 | 16,44464663 |
| 3 | 6 | 20,08553669 |
| 3,2 | 6,4 | 24,53252981 |
| 3,4 | 6,8 | 29,96409944 |
| 3,6 | 7,2 | 36,59823348 |
| 3,8 | 7,6 | 44,701183 |
| 4 | 8 | 54,59814775 |

Как видно из полученных таблиц результатов, точность решения в 0.0001 при решении методом Рунге-Кутта с непостоянным шагом достигается всего за 10 шагов, в то время, когда для достижения этой же точности при решении методом Рунге-Кутта 5-го порядка с постоянным шагом требуется около 100 шагов.

Сравнивая полученные результаты с результатами работы программы “Adams3.exe”, приходим к выводу, что неявная схема Адамса третьего порядка достаточно эффективна при численном решении задачи Коши (быстрота, высокая точность решения), однако по своим характеристикам она уступает более совершенным методам, применяющимися в различных математических пакетах.

**Заключение**

Результатом выполнения курсового проекта является готовый программный продукт, позволяющий решать задачу Коши для системы дифференциальных уравнений при помощи неявной схемы Адамса 3-го порядка, демонстрирующий возможности численного решения поставленной задачи с заданной степенью точности.

Готовый программный продукт может найти широкое применение при решении многих прикладных технических программ, а в частности, эффективно использование применённой схемы Адамса 3-го порядка для решения так называемых “жёстких” систем дифференциальных уравнний, для которых существует лишь численное решение.

Данная программа решает заданную пользователем систему дифференциальных уравнений с указанной точностью за минимальный промежуток времени. При этом пользователю предоставляется возможность визуально оценить неточность решения, сравнивая графики полученного и точного решений.

К достоинствам программы можно отнести также удобный пользовательский интерфейс, возможность ввода пользовательских систем дифференциальных уравнений, а также высокая стабильность работы. Однако имеются и некоторые недостатки. К недостаткам программы можно отнести: критичность к вводимым пользователем функций, отсутсвие обработки исключительных событий. Это, естественно, ограничивает возможности программы.

**Литература**

1. Архангельский А.Я. Программирование в С++ Builder 6. – М.: ЗАО “Издательство БИНОМ”, 2002. – 360 с.

2. Калиткин Н.Н. Численные методы. ⎯ М.: Наука, 1978. ⎯ 512 с.

3. Самарский А.А., Гулин А.В. Численные методы. ⎯ М.: Наука, 1989. – 432с.

4. Синицын А.К., Навроцкий А.А. Алгоритмы вычислительной математики. - Мн.: БГУИР,2002. – 80 с: ил.

5. Синицин А.К. Программирование алгоритмов в среде Builder C++. –Мн.: БГУИР, 2004. – 90 с.: ил.

6. Страуструп Бьерн. Язык программирования C++. –М.: ЗАО “Издательство БИНОМ”, 2002. – 1099c.:ил.

7. Шилд Г. Программирование на Borland C++ для профессионалов— М.:ООО “попурри” ,1999. – 800c.:ил.

**Приложения**

**Приложение 1**

**Блок-схема алгоритма**



**Блок-схема решения задачи Коши неявной схемой Адамса 3-го порядка.**



**Блок-схема алгоритма преобразования строки в обратную польскую запись:**



**Блок-схема вычисления функций:**



**Приложение 2**

**Листинг программы**

**Главная программа (Unit1.cpp):**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include "math.h"

#include "stdio.h"

#include "Unit3.h"

#include "Unit5.h"

#include "fstream.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

char \*opz(char \*); // ф-ия преобразования в обратную польскую запись;

double fpr(char \*str,double u, double v,double x); // обратный ход польской

int p=1,s=1,j=1,o=0; // записи;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N5Click(TObject \*Sender)

{

Form1->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

Form1->Close();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N7Click(TObject \*Sender)

{

Form2->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N2Click(TObject \*Sender) // очистка формы

{

Edit1->Clear();

Edit2->Clear();

Edit3->Clear();

Edit4->Clear();

Edit5->Clear();

Edit6->Clear();

Edit7->Clear();

Edit8->Clear();

Edit9->Clear();

Memo1->Clear();

Series1->Clear();

Series2->Clear();

Series3->Clear();

Series4->Clear();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

Edit1->Text="10";

Edit2->Text="1";

Edit3->Text="2";

Edit4->Text="4";

Edit5->Text="0,0001";

Edit6->Text="4";

Edit7->Text=FloatToStrF(exp(2),ffFixed,20,18);

Memo1->Text="результаты программы";

Button1->Show();

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender) //обработка события нажатия кнопки «выполнить»

{

//---------------------------------------------------------------------------

int nx,np,k,i,n;

double a,b,e,h,d,de,z,x;

double y[2],yp[2],f[2],fm[2],fp[2],fp1[2],fp2[2];

unsigned long int time=GetTickCount();

unsigned long int time1=0;

a=StrToFloat(Edit3->Text);

b=StrToFloat(Edit4->Text);

e=StrToFloat(Edit5->Text);

nx=StrToInt(Edit1->Text);

np=StrToInt(Edit2->Text);

Memo1->Clear();

Memo1->Lines->Add("Входные данные:");

Memo1->Lines->Add("");

Memo1->Lines->Add("du/dx="+Edit8->Text+";");

Memo1->Lines->Add("dv/dx="+Edit9->Text+";");

Memo1->Lines->Add("Интервал: ["+Edit3->Text+";"+Edit4->Text+"]");

Memo1->Lines->Add("Допустимая погрешность: е="+Edit5->Text);

Memo1->Lines->Add("Начальные условия:");

Memo1->Lines->Add("u="+Edit6->Text);

Memo1->Lines->Add("v="+Edit7->Text);

Memo1->Lines->Add("Количество шагов сетки: "+Edit1->Text);

Memo1->Lines->Add("Шаг вывода: "+Edit2->Text);

Memo1->Lines->Add("");

Memo1->Lines->Add("");

char \*u1 =(char \*)malloc(strlen(Edit8->Text.c\_str())+1);

char \*v1 =(char \*)malloc(strlen(Edit9->Text.c\_str())+1);

strcpy(u1,Edit8->Text.c\_str());

strcpy(v1,Edit9->Text.c\_str());

char \*u =(char \*)malloc(strlen(u1)+1); //динамическое выделение памяти

char \*v =(char \*)malloc(strlen(v1)+1);

strcpy(u,opz(&(u1[0]))); // преобразование в обратную польскую запись

strcpy(v,opz(&(v1[0])));

do {

h=(b-a)/nx;

x=a;

y[0]=StrToFloat(Edit6->Text);

y[1]=StrToFloat(Edit7->Text);

if(np!=0&&s==0){

Memo1->Lines->Add("Результаты:");

Memo1->Lines->Add(" x | u(x) | точное | разн. | v(x) | точное | разн. | ");

Memo1->Lines->Add("-----------------------------------------------------");

Memo1->Lines->Add(FloatToStrF(x,ffFixed,5,3)+" "+FloatToStrF(y[0],ffFixed,8,4)+" "+FloatToStrF(2\*x,ffFixed,8,4)+" "+FloatToStrF(y[0]-2\*x,ffFixed,8,4)+" "+FloatToStrF(y[1],ffFixed,8,4)+" "+FloatToStrF(exp(x),ffFixed,8,4)+" "+FloatToStrF(y[1]-exp(x),ffFixed,8,4));

}

Series1->Clear();

Series2->Clear();

Series3->Clear();

Series4->Clear();

Series1->AddXY(x,y[0]);

Series2->AddXY(x,2\*x);

Series3->AddXY(x,y[1]);

Series4->AddXY(x,exp(x));

fm[0]=fpr(u,y[0],y[1],x);

fm[1]=fpr(v,y[0],y[1],x);

for(i=0;i<2;i++)

{

yp[i]=y[i]+h/2\*fm[i];

}

x=x+h/2;

fp1[0]=fpr(u,yp[0],yp[1],x);

fp1[1]=fpr(v,yp[0],yp[1],x);

for(i=0;i<2;i++)

{

yp[i]=y[i]+h/2\*fp1[i];

}

fp2[0]=fpr(u,yp[0],yp[1],x);

fp2[1]=fpr(v,yp[0],yp[1],x);

for(i=0;i<2;i++)

{

yp[i]=y[i]+h\*fp2[i];

}

x=x+h/2;

fp[0]=fpr(u,yp[0],yp[1],x);

fp[1]=fpr(v,yp[0],yp[1],x);

for(i=0;i<2;i++)

{

yp[i]=y[i]+h\*(fm[i]+2\*fp1[i]+2\*fp2[i]+fp[i])/6;

}

fp[0]=fpr(u,yp[0],yp[1],x);

fp[1]=fpr(v,yp[0],yp[1],x);

for(n=2;n<=nx;n++)

{

for(i=0;i<2;i++)

{

y[i]=yp[i]+h\*(1.5\*fp[i]-0.5\*fm[i]);

};

x=x+h;

f[0]=fpr(u,y[0],y[1],x);

f[1]=fpr(v,y[0],y[1],x);

k=0;

do

{

k=k+1;

de=0;

for(i=0;i<2;i++)

{

z=yp[i]+h\*(5\*f[i]+8\*fp[i]-fm[i])/12;

d=fabs(z-y[i]);

y[i]=z;

if(d>de) de=d;

};

f[0]=fpr(u,y[0],y[1],x);

f[1]=fpr(v,y[0],y[1],x);

} while(de>e);

for(i=0;i<2;i++)

{

yp[i]=y[i];

fm[i]=fp[i];

fp[i]=f[i];

}

Series1->AddXY(x,y[0]); //вывод графиков функций

Series2->AddXY(x,2\*x);

Series3->AddXY(x,y[1]);

Series4->AddXY(x,exp(x));

if((fmod(n,np)==0)&&s==0) { //вывод результатов

Memo1->Lines->Add(FloatToStrF(x,ffFixed,5,3)+" "+FloatToStrF(y[0],ffFixed,8,4)+" "+FloatToStrF(2\*x,ffFixed,8,4)+" "+FloatToStrF(y[0]-2\*x,ffFixed,8,4)+" "+FloatToStrF(y[1],ffFixed,8,4)+" "+FloatToStrF(exp(x),ffFixed,8,4)+" "+FloatToStrF(y[1]-exp(x),ffFixed,8,4));

p=1;

o=1;

}

else o=0;

}

nx=nx\*2;

np=np\*2;

time1=GetTickCount();

if (o==1) {

Memo1->Lines->Add("------------------------------------------------------");

Memo1->Lines->Add("Время выполнения:"+FloatToStrF((time1-time)/1000.,ffFixed,6,3)+"мс");

}

if(CheckBox1->Checked) y[1]=exp(x);

} while(fabs(y[1]-exp(x))>e);

j++;

s=1;

if(p==1&&(fmod(j,2)==0))

{

Memo1->Lines->Add("Рекомендуемое значение шага сетки :"+FloatToStrF(nx/2,ffFixed,6,0));

Edit1->Text=FloatToStrF(nx/2,ffFixed,5,0);

Edit2->Text=FloatToStrF(np/2,ffFixed,5,0);

s=0;

p=0;

}

free(u); // освобождение памяти

free(v);

free(u1);

free(v1);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N4Click(TObject \*Sender) //Сохранение в файл

{

SaveDialog1->Title="Save File";

if (SaveDialog1->Execute())

{

Memo1->Lines->SaveToFile(SaveDialog1->FileName);

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::N3Click(TObject \*Sender) // Загрузка из файла функций

{

if(OpenDialog1->Execute())

{

FILE \*fl;

fl=fopen(OpenDialog1->FileName.c\_str(),"r");

char ch=getc(fl);

char str[30];

str[0]='\0';

int k=0;

while (ch!=EOF)

{

if(ch=='=') { k++;

while (ch!=';'){ ch=getc(fl);

int n=strlen(str);

str[n]=ch;

str[n+1]='\0';

}

switch (k)

{

case 1: Edit8->Text=str; str[0]='\0'; break;

case 2: Edit9->Text=str; break;

}

}

ch=getc(fl);

}

fclose(fl);

}

}

//---------------------------------------------------------------------------

**Модуль преобразования строки в обратную польскую запись (Unit3.cpp):**

//---------------------------------------------------------------------------

#pragma hdrstop

#include "Unit3.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#include<stdio.h>

#include<stdlib.h>

#include<conio.h>

struct st {

char c;struct st \*next;

};

struct st \*push(struct st \*,char);

char DEL(struct st \*\*);

int PRIOR(char);

char\* opz(char \*a)

{

struct st \*OPERS=NULL;

char \*outstring= new char [30]; // динамическое выделение памяти

int k,point;

k=point=0;

while((\*(a+k)!='\0')&&(\*(a+k)!='=')){

if(\*(a+k)==')'){

while((OPERS->c)!='(')

outstring[point++]=DEL(&OPERS);

DEL(&OPERS);

}

if((\*(a+k)>='a'&&(\*(a+k))<='z')||(\*(a+k)>='1'&&(\*(a+k))<='9'))

outstring[point++]=\*(a+k);

if(a[k]=='(')

OPERS=push(OPERS,'(');

if(\*(a+k)=='+'||\*(a+k)=='-'||\*(a+k)=='/'||\*(a+k)=='\*'||\*(a+k)=='^'){

if(OPERS==NULL)

OPERS=push(OPERS,\*(a+k));

else

if(!PRIOR(OPERS->c))

OPERS=push(OPERS,\*(a+k));

else{

while((OPERS!=NULL)&&(PRIOR(OPERS->c)>=PRIOR(\*(a+k))))

outstring[point++]=DEL(&OPERS);

OPERS=push(OPERS,\*(a+k));

}

}

k++;

}

while(OPERS!=NULL)

outstring[point++]=DEL(&OPERS);

outstring[point]='\0';

return outstring;

}

struct st \*push(struct st \*HEAD,char a) /\* Функция записывает в стек,на веpшину котоpого указывает HEAD,символ a.

Возвpащает указатель на новую веpшину стека\*/

{

struct st \*PTR;

PTR=new st ();

PTR->c=a;

PTR->next=HEAD;

return PTR;

}

char DEL(struct st \*\*HEAD){ /\* функция удаляет символ с веpшины стека. Возвpащает удаляемый символ.

Изменяет указатель на веpшину стека\*/

struct st \*PTR;

char a;

if(\*HEAD==NULL)

return '\0';

PTR=\*HEAD;

a=PTR->c;

\*HEAD=PTR->next;

free(PTR);

return a;

}

int PRIOR(char a) //функция возвpащает пpиоpитет аpифметической опеpации

{

switch(a){

case '^':

return 4;

case '\*':

case '/':

return 3;

case '-':

case '+':

return 2;

case '(':

return 1;

}

}

**Модуль расчёта функции, записанной в постфиксной форме (Unit5.cpp):**

//---------------------------------------------------------------------------

#pragma hdrstop

#include "Unit5.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#include<conio.h>

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<string.h>

double fpr(char \*str,double u, double v,double x)

{

int n,i,d=0;

double th[30],g[30];

n=strlen(str) ;

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

{

switch (\*(str+i))

{

case 'x': \*(th+i)=x; break;

case 'u': \*(th+i)=u; break;

case 'v': \*(th+i)=v; break;

case 'e': \*(th+i)=exp(1); break;

case '1':

case '2':

case '3':

case '4':

case '5':

case '6':

case '7':

case '8':

case '9':

case '0': char p[1]; p[0]=str[i]; th[i]=atoi(p); break;

}

}

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

{

if(\*(str+i)=='x'||\*(str+i)=='v'||\*(str+i)=='u'||\*(str+i)=='e'||\*(str+i)=='1'||\*(str+i)=='2'||\*(str+i)=='3'||\*(str+i)=='4'||\*(str+i)=='5'||\*(str+i)=='6'||\*(str+i)=='7'||\*(str+i)=='8'||\*(str+i)=='9')

{

\*(g+d)=\*(th+i);

d++;

}

else {

switch (\*(str+i))

{

case '-': \*(g+d-2)=\*(g+d-2)-\*(g+d-1); break;

case '+': \*(g+d-2)=\*(g+d-2)+\*(g+d-1); break;

case '/': \*(g+d-2)=\*(g+d-2)/(\*(g+d-1)); break;

case '\*': \*(g+d-2)=\*(g+d-2)\*(\*(g+d-1)); break;

case '^': \*(g+d-2)=pow(\*(g+d-2),\*(g+d-1)); break;

};

d--;

}

}

return \*g;

}

**Приложение 3**

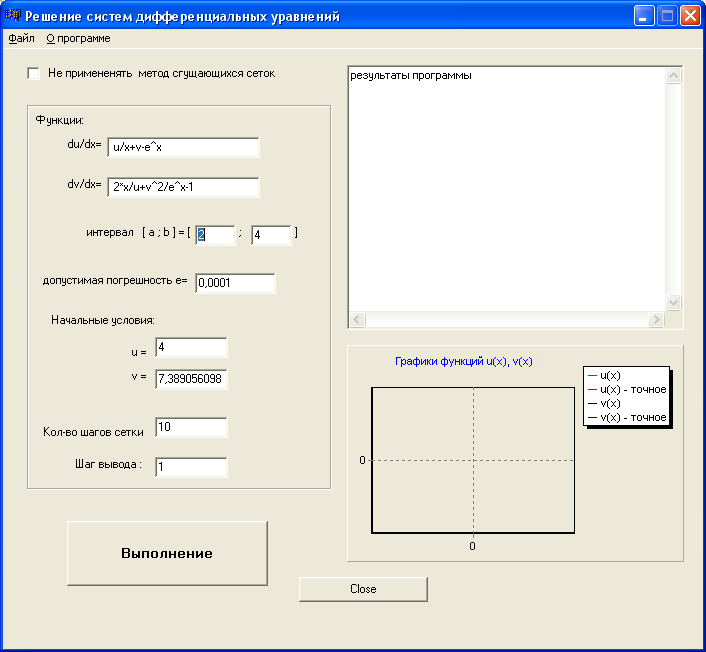


Рис 1. Общий вид программы

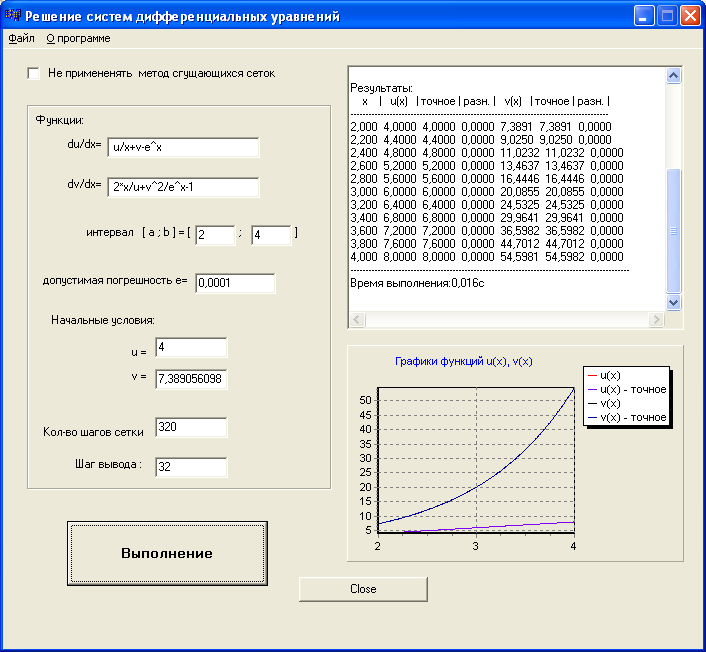


Рис 2. Организация решения системы

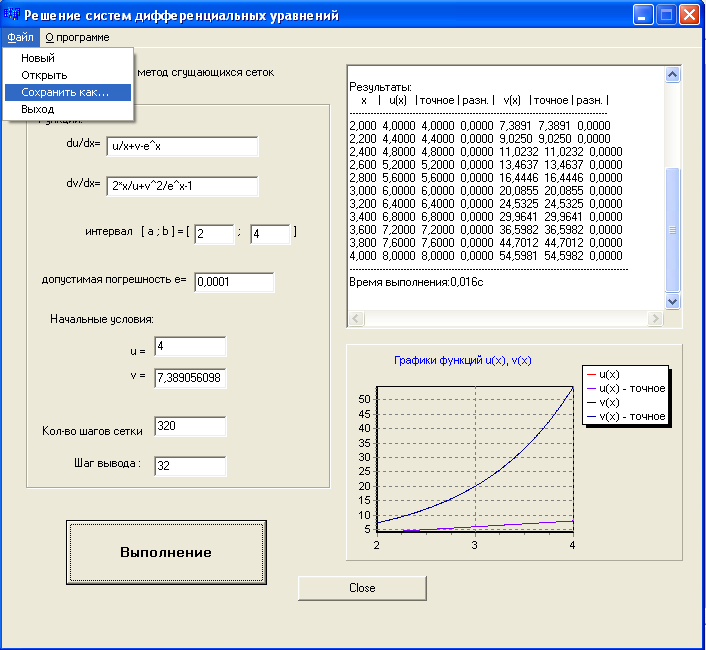


Рис 3. Организация меню