Министерство образования республики Беларусь

Учреждение образования

Белорусский государственный университет информатики и радиоэлектроники

Факультет информационных технологий и управления

Кафедра вычислительных методов и программирования

Дисциплина: Основы алгоритмизации и программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему:

“Решение задачи Коши методом Рунге-Кутта 4 порядка”

 Студентка: гр.020604 Хилько П.Г.

Руководитель: д.ф.-м.н., профессор Синицин А.К.

Минск 2011

# *Содержание*

1. Задание на курсовую работу,
2. Содержание,
3. Введение,
4. Описание алгоритмов решения поставленной задачи,
5. Описание тестовой задачи и результатов работы программы,
6. Заключение,
7. Литература,
8. Текст программы,
9. Интерфейс.

***Задание на курсовую работу***

Соотношение вида

, (1)

где  – некоторая функция независимой переменной , функции  и ее производных , , …, , называется обыкновенным дифференциальным уравнением -го порядка. Решить уравнение – значит найти функцию , превращающую равенство (1) в тождество. Существует понятие *общего* и *частного* решения этого дифференциального уравнения. *Общее* решение (общий интеграл) – это формула, дающая все решения данного уравнения. Обычно общее решение обыкновенного дифференциального уравнения -го порядка (1) зависит от  постоянных , которые могут выбираться произвольно. Решение, которое не зависит от произвольных постоянных, называется *частным* решением дифференциального уравнения (частным интегралом). График каждого частного решения называется *интегральной кривой*. Чаще всего для обыкновенного дифференциального уравнения (1) формулируется так называемая задача Коши, когда дополнительно к уравнению (1) задают значения функции и ее производных до -го порядка в некоторой точке . Эти дополнительные данные называются начальными условиями. Наличие начальных условий позволяет получить частное решение дифференциального уравнения.

 Процесс решения дифференциального уравнения называется его *интегрированием*. Интегрирование дифференциального уравнения вовсе не означает, что этот процесс сводится к вычислению интеграла. Если же решение дифференциального уравнения действительно свелось к вычислению интеграла, то говорят, что уравнение решено *в квадратурах*.

 Методы решения дифференциальных уравнений бывают точные, приближенные и численные. Точные методы дают решение, которое можно выразить через элементарные функции. Получить точное решение дифференциального уравнения можно не всегда. Например, решение уравнения  не выражается через элементарные функции. Приближенные методы дают решение в виде некоторой последовательности функций , сходящейся к решению  при . Численные методы дают решение в виде таблицы значений функции .

***Введение***

 Обыкновенные дифференциальные уравнения (ОДУ) широко используются для математического моделирования процессов и явлений в различных областях науки и техники. Переходные процессы в радиотехнике, кинетика химических реакций, динамика биологических популяций, движение космических объектов, модели экономического развития исследуются с помощью ОДУ. Поэтому решение обыкновенных дифференциальных уравнений занимает одно из важнейших мест среди прикладных задач физики, электроники, химии и техники.

   В дифференциальное уравнение n-го порядка в качестве неизвестных величин входят функция y(x) и ее первые n производных по аргументу x:

    f( x, y, y1, ...  y(n) )=0 (2)

   Из теории ОДУ известно, что уравнение (2) эквивалентно системе n уравнений первого порядка

fk(x, y1, y1’ ,y2 ,y2 ’, ... ,yn ,yn ’)=0.      (3)

где k=1, ... , n.

Уравнение (2) и эквивалентная ему система (3) имеют 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 (2) в некоторой точке xo должны быть заданы начальные условия, т.е. значения функции y(x) и ее производных

               y(x0)=y0’ ,   y’(x0)=y10, ... , $y^{(n-1)(}$x0)=yn-1,0.

Для системы ОДУ типа (3) начальные условия задаются в виде

y1(x0)=y10 ,     y2(x0)=y20, ... , $y\_{n}$ (x0)=yn0.           (4)

   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 є [x0 ,xk], то такие условия могут быть заданы как на границах, так и внутри интервала. Минимальный порядок ОДУ, для которых может быть сформулирована граничная задача, равен двум.

   Третий тип задач для ОДУ – это задачи на собственные значения. Такие задачи отличаются тем, что кроме искомых функций y(x) и их производных в уравнения входят дополнительно m неизвестных параметров l1,l2,¼, хm, которые называются собственными значениями. Для единственности решения на интервале [x0,xk] необходимо задать m+n граничных условий. В качестве примера можно назвать задачи определения собственных частот, коэффициентов диссипации, структуры электромагнитных полей и механических напряжений в колебательных системах, задачи нахождения фазовых коэффициентов, коэффициентов затухания, распределения напряженностей полей волновых процессов и т.д.

   К численному решению ОДУ приходится обращаться, когда не удается построить аналитическое решение задачи через известные функции. Хотя для некоторых задач численные методы оказываются более эффективными даже при наличии аналитических решений. Cамое большое распространение из всех численных методов решения дифференциальных уравнений с помощью ЭВМ получил метод Рунге-Кутта 4-го порядка.

***Описание алгоритмов решения поставленной задачи***

Рассмотрим для начала **Метод Рунге-Кутта 2-го порядка**.

Этот метод состоит в последовательных расчетах по формулам

 ,

,

,

начиная с точки . Метод Рунге-Кутта 2-го порядка имеет погрешность .

Рассмотрим этот метод более конкретно.

 (5)

Используя в (6) формулу средних, получим

 (6)

Уравнение разрешено явно относительно , однако в правой части присутствует неизвестное значение  в середине отрезка . Для решения этого уравнения существует следующий способ. Вначале по явной схеме



рассчитывают  (предиктор):



После этого рассчитывают  по формуле (6) (корректор). В результате схема оказывается явной и имеет второй порядок. Алгоритм:

****

Схема Рунге-Кутта 2-го порядка

Метод Рунге-Кутта 4-го порядка

 Используя в



формулу Симпсона, получим

. (8)

 Можно по-разному реализовать расчет неявного по  уравнения (8), однако наибольшее распространение получил следующий способ. Делают предиктор вида

 

затем корректор по формуле

 

Алгоритм метода Рунге-Кутта 4 порядка:

***Описание тестовой задачи и результатов работы программы***

Рассмотрим решение задачи при помощи данной программы.

Пример:

$$\left\{\begin{array}{c}\dot{x}=3x-y-z \\\dot{y}=-x+5y-z\\\dot{z}=x-y+3z \end{array}\right.$$

при x(0)=7,

 y(0)=3,

 z(0)=5.

 Чтобы решить нашу систему мы должныпоступить следующим образом:

1. Мы должны внести в программу систему функций.
2. В интерфейсе изменить начальные условия, а именно:
* x(0), y(0), z(0).
* начальную точку конечную, а также число шагов.



3. Когда все начальные условия введены, для того, чтобы получить ответ, мы должны нажать на кнопку «Решить». В результате появятся графики трёх функций: x(t), y(t), z(t).



***Заключение (оценка работы и возможностей программы)***

Проблема повышения качества вычислений, как несоответствие между желаемым и действительным, существует и будет существовать в дальнейшем. Ее решению будет содействовать развитие информационных технологий, которое заключается как в совершенствовании методов организации информационных процессов, так и их реализации с помощью конкретных инструментов – сред и языков программирования.

 Итогом работы можно считать созданную функциональную модель численного решения обыкновенных дифференциальных уравнений методом Рунге-Кутта 4 порядка.

***Литература***

1. Ж. А. Черняк «Электронный учебно-методический комплекс по высшей математике».
2. Калиткин Н.Н. «Численные методы». 1978.
3. Самарский А.А., Гулин А.В. «Численные методы». 1989.
4. Колосов С.В. «Программирование в среде Delphi: Учеб. пособие» 2005.
5. Письменный Д.Т. «Конспект лекций по высшей математике: полный курс». 2009.

***Текст программы***

unit Unit1;

interface

uses

 Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

 Dialogs, Buttons, StdCtrls, TeEngine, TeeFunci, Series, ExtCtrls,

 TeeProcs, Chart, Unit2;

type

 TForm1 = class(TForm)

 GroupBox1: TGroupBox;

 GroupBox2: TGroupBox;

 GroupBox3: TGroupBox;

 Button1: TButton;

 Chart1: TChart;

 Series1: TLineSeries;

 TeeFunction1: TAddTeeFunction;

 Label1: TLabel;

 Label4: TLabel;

 Label5: TLabel;

 Label6: TLabel;

 Edit1: TEdit;

 Edit2: TEdit;

 Edit3: TEdit;

 BitBtn1: TBitBtn;

 Edit5: TEdit;

 GroupBox4: TGroupBox;

 Series2: TLineSeries;

 Label2: TLabel;

 Edit4: TEdit;

 Edit6: TEdit;

 Label3: TLabel;

 Series3: TLineSeries;

 Label7: TLabel;

 Edit7: TEdit;

 procedure FormCreate(Sender: TObject);

 procedure Button1Click(Sender: TObject);

 private

 { Private declarations }

 public

 { Public declarations }

 end;

var

 Form1: TForm1;

implementation

{$R \*.dfm}

procedure TForm1.FormCreate(Sender: TObject);

begin

 Edit1.Text:='0';

 Edit2.Text:='5';

 Edit3.Text:='10';

 Edit4.Text:='7';

 Edit6.Text:='3';

 Edit7.Text:='5';

 Edit5.Clear;

end;

function f0(Vars:TVarsArray):extended;

begin

 Result:=3\*vars[0]-vars[1]-vars[2];

end;

function f1(Vars:TVarsArray):extended;

begin

 Result:=-vars[0]+5\*vars[1]-vars[2];

end;

function f2(Vars:TVarsArray):extended;

begin

 Result:=vars[0]-vars[1]+3\*vars[2];

end;

procedure TForm1.Button1Click(Sender: TObject);

var I: Integer;

 FunArray: TFunArray; // массив функций

 tn: Extended; // начальная точка по независимой координате

 tk: Extended; // конечная точка по независимой координате

 Shag: Integer; // число шагов по независимой координате

 y: Ty; // вектор начальных значений

 Res: TResArray; // матрица результатов включая независ. переменную

begin

 // Создаем вектор функций:

 SetLength(FunArray,3);

 FunArray[0]:=f0;

 FunArray[1]:=f1;

 FunArray[2]:=f2;

 // Задаем интервал и число шагов:

 tn:=StrToInt(Edit1.Text);

 tk:=StrToInt(Edit2.Text);

 Shag:=StrToInt(Edit3.Text);

 // Задаем начальные условия:

 SetLength(y,3);

 y[0]:=StrToInt(Edit4.Text);

 y[1]:=StrToInt(Edit6.Text);

 y[2]:=StrToInt(Edit7.Text);

 Edit5.Text:=FloatToStr(abs(Runge\_Kutt(FunArray, tn, tk, Shag, y, Res)));

 Chart1.Series[0].Clear;

 For I:=0 to Shag do

 Chart1.Series[0].AddXY(Res[0,I], Res[1,I], '',clTeeColor);

 Chart1.Series[1].Clear;

 For I:=0 to Shag do

 Chart1.Series[1].AddXY(Res[0,I], Res[2,I], '',clTeeColor);

 Chart1.Series[2].Clear;

 For I:=0 to Shag do

 Chart1.Series[2].AddXY(Res[0,I], Res[3,I], '',clTeeColor);

end;end.

unit Unit2;

interface

 type

 TVarsArray = array of Extended; // вектор переменных включая независимую

 Ty = array of Extended; // вектор начальных значений

 TFunArray = array of function(VarsArray: TVarsArray ):Extended;

 // вектор функций

 TResArray = array of array of Extended; // матрица результатов

 TCoefsArray = array of Extended; // вектор коэффициетов метода

 function Runge\_Kutt( // метод Рунге-Кутта

 FunArray: TFunArray; // массив функций

 tn: Extended; // начальная точка по независимой координате

 tk: Extended; // конечная точка по независимой координате

 Shag: Integer; // число шагов по независимой координате

 y: Ty; // вектор начальных значений

 var Res: TResArray // матрица результатов включая независ. переменную

 ):Extended;

 // возвращаемое значение - погрешность

implementation

Function Runge\_Kutt( // метод Рунге-Кутта

 FunArray: TFunArray; // массив функций

 tn: Extended; // начальная точка по независимой координате

 tk: Extended; // конечная точка по независимой координате

 Shag: Integer; // число шагов по независимой координате

 y: Ty; // вектор начальных значений

 var Res: TResArray // матрица результатов включая независ. переменную

 ):extended; // возвращаемое значение - погрешность

var

 N: Word; // число уравнений

 NachUsl: Word; // число начальных условий

 h: Extended; // шаг разбиения

 Vars: TVarsArray; // вектор переменных включая независимую

 Vars2,Vars3,Vars4: TVarsArray; // значения перем. для 2-4 коэф.

 Coefs1: TCoefsArray; // вектор 1-ыx коэффициентов в методе

 Coefs2: TCoefsArray; // вектор 2 коэффициентов в методе

 Coefs3: TCoefsArray; // вектор 3 коэффициентов в методе

 Coefs4: TCoefsArray; // вектор 4 коэффициентов в методе

 I: Integer; // счетчик цикла по иттерациям

 J: Word; // индекс коэф.-тов метода

 K: Integer; // счетчик прочих циклов

begin

 N:=Length(FunArray); // узнаем число уравнений

 NachUsl:=Length(y); // узнаем число начальных условий

 h:=(tk-tn)/Shag; // находим величину шага разбиений

 SetLength(Res,N+1,Shag+1); // задаем размер матрицы ответов с незав. перем.

 SetLength(Vars,N+1); // число переменных включая независимую

 SetLength(Vars2,N+1); // число переменных для 2-го коэф. включая независимую

 SetLength(Vars3,N+1); // число переменных для 3-го коэф. включая независимую

 SetLength(Vars4,N+1); // число переменных для 4-го коэф. включая независимую

 SetLength(Coefs1,N); // число 1-ыx коэф. метода по числу уравнений

 SetLength(Coefs2,N); // число 2-ыx коэф. метода по числу уравнений

 SetLength(Coefs3,N); // число 3-иx коэф. метода по числу уравнений

 SetLength(Coefs4,N); // число 4-ыx коэф. метода по числу уравнений

 // Начальные значения переменных:

 Vars[0]:=tn;

 For K:=0 to NachUsl-1 do Vars[K+1]:=y[K];

 For J:=0 to N do Res[J,0]:=Vars[J]; // первая точка результата

 For I:=0 to Shag-1 do // начало цикла иттераций

 begin

 For J:=0 to N-1 do Coefs1[J]:=FunArray[J](Vars)\*h; // 1-й коэфф.

 // Находим значения переменных для второго коэф.

 Vars2[0]:=Vars[0]+h/2;

 For K:=1 to N do Vars2[K]:=Vars[K]+Coefs1[K-1]/2;

 For J:=0 to N-1 do Coefs2[J]:=FunArray[J](Vars2)\*h; // 2-й коэф.

 {// Находим значения переменных для третьго коэф.

 Vars3[0]:=Vars[0]+h/2;

 For K:=1 to N do Vars3[K]:=Vars[K]+Coefs2[K-1]/2;

 For J:=0 to N-1 do Coefs3[J]:=FunArray[J](Vars3)\*h; // 3 коэфф.

 // Находим значения переменных для 4 коэф.

 Vars4[0]:=Vars[0]+h;

 For K:=1 to N do Vars4[K]:=Vars[K]+Coefs3[K-1];

 For J:=0 to N-1 do Coefs4[J]:=FunArray[J](Vars4)\*h; // 4 коэфф.

 // Находим новые значения переменных включая независимую}

 Vars[0]:=Vars[0]+h;

 For K:=1 to N do

 Vars[K]:=Vars[K]+

 (1/6)\*(Coefs1[K-1]+2\*(Coefs2[K-1]+Coefs3[K-1])+Coefs4[K-1]);

 // Результат иттерации:

 For J:=0 to N do Res[J,I+1]:=Vars[J];

 end; // конец итераций

 Result:=h\*h\*h\*h\*h\*res[1,1]; // код ошибки погрешность

end;

end.

***Интерфейс***

