|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Министерство образования и науки Российской Федерации | | | | |
| Федеральное государственное бюджетное образовательное учреждение  Высшего образования  «Кузбасский государственный технический университет им.Т.Ф.Горбачева» | | | | |
| Кафедра прикладных информационных технологий | | | | |
| Дисциплина: | Информатика и программирование | | | |
| Направление подготовки: | 09.03.03 Прикладная информатика | | | |
|  | | | | |
| Лабораторная работа № | | 4 | на тему: Решение систем дифференциальных уравнений | |
|  | | |
|  | | | Выполнил студ. группы | Минлигареев М.А.  ПИб-152 |
| Принял: | Ещин Е.К. |
|  | |
| Кемерово 2016 | | | | |

**Содержание**

[Введение 3](#_Toc450313153)

[1. Описание программы 4](#_Toc450313154)

[1.1 Общие сведения 4](#_Toc450313155)

[1.2 Функциональное назначение 4](#_Toc450313156)

[1.3 Описание логической структуры 4](#_Toc450313157)

[1.4 Используемые технические средства 14](#_Toc450313158)

[1.5 Вызов и загрузка 14](#_Toc450313159)

[1.6 Входные данные 16](#_Toc450313160)

[1.7 Выходные данные 16](#_Toc450313161)

[Заключение 17](#_Toc450313162)

# Введение

Системой дифференциальных уравнений называется совокупность уравнений, в каждое из которых входят независимая переменная, искомые функции и их производные. Всегда предполагается, что число уравнений равно числу неизвестных функций.

В данной лабораторной работе под решением дифференциальных уравнений, представленных на рисунке 1, понимается не использование аналитических методов, а использование численных методов для реальных объектов, так как они имеют большое количество нелинейностей.



Рисунок 1 – Общий вид дифференциального уравнения

Решение дифференциального уравнения заключается в том, чтобы установить количественную связь в виде «аргумент – функция» с использованием записи Леонарда Эйлера, которая представлена на рисунке 2.



Рисунок – Общий вид дифференциального уравнения по Эйлеру

Из приведённого выше уравнения можно сделать вывод, представленный на рисунке 2.



Рисунок – Трансформированное уравнение Эйлера

Задача лабораторной работы заключается в том, чтобы определить и отобразить фазовые траектории при оптимальном по быстродействию управлении объектом второго порядка опираясь на использование методов решения дифференциальных уравнений второго порядка, соответственно.

# 1. Описание программы

1.1 Общие сведения

Программный продукт носит название «Differential\_Equations» (от англ. «Дифференциальные уравнения») и имеет расширение «.exe».

Программное обеспечение, необходимое для функционирования программного продукта, включает в себя:

* операционную систему Windows любой версии;

Программный продукт написан на языке программирования Delphi в инструментальной среде разработки Delphi XE7 от компании Embarcadero.

1.2 Функциональное назначение

Программный продукт, используя полученные с клавиатуры данные от пользователя выполняет вычисление и построение фазовых траекторий при оптимальном по быстродействию управлении объектом второго порядка опираясь на использование методов решения дифференциальных уравнений второго порядка, соответственно.

1.3 Описание логической структуры

Программа состоит из трёх программных модулей.

Алгоритм работы основного модуля программы представлен на рисунке 4.

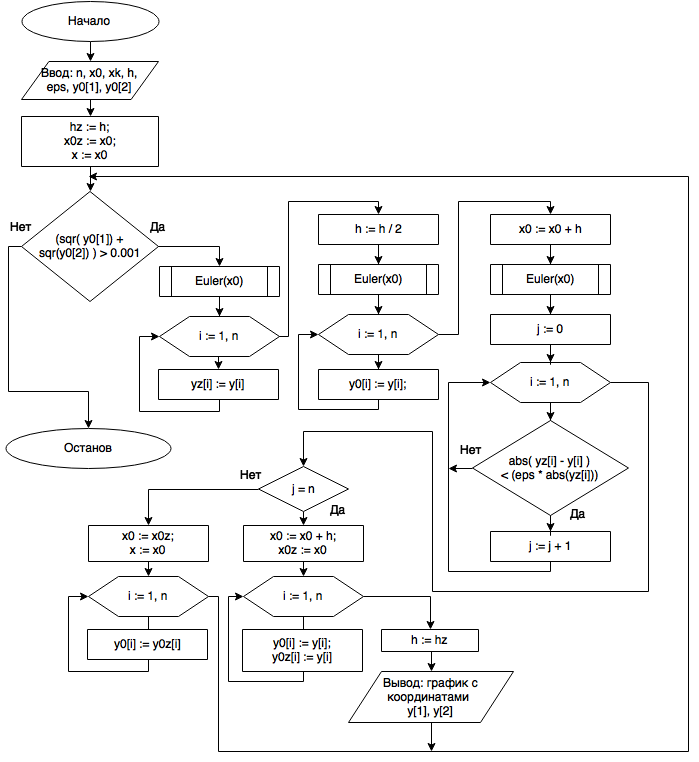


Рисунок 4 – Алгоритм работы основного модуля программы

Алгоритм работы модуля Euler представлен на рисунке 5.

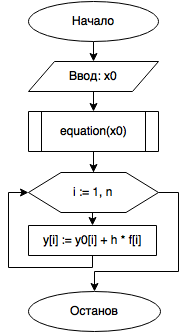


Рисунок 5 – Алгоритм работы модуля Euler

Алгоритм работы модуля equation представлен на рисунке 6.

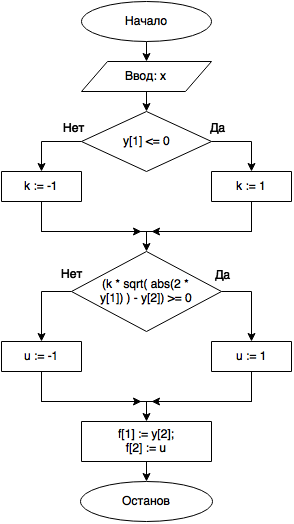


Рисунок 6 – Алгоритм работы модуля equation

Структура программы включает в себя:

{ Обозначения :

S | V | F/m | t

X1 | X2 | U | t

Y[1] | Y[2] | U | x0

dX1/dt=F[1] - правая часть 1-го уравнения

dX2/dt=F[2] - правая часть 2-го уравнения}

unit Diff\_Equations;

interface

uses

Winapi.Windows, Winapi.Messages, System.SysUtils, System.Variants, System.Classes, Vcl.Graphics,

Vcl.Controls, Vcl.Forms, Vcl.Dialogs, Vcl.StdCtrls, Vcl.Imaging.pngimage,

Vcl.ExtCtrls, VclTee.TeeGDIPlus, VCLTee.Series, VCLTee.TeEngine,

VCLTee.TeeProcs, VCLTee.Chart;

type

TForm1 = class(TForm)

Panel1: TPanel;

Label3: TLabel;

Label1: TLabel;

Label2: TLabel;

Edit1: TEdit;

Edit2: TEdit;

Panel2: TPanel;

Label5: TLabel;

Edit3: TEdit;

Label4: TLabel;

Edit4: TEdit;

Label6: TLabel;

Edit5: TEdit;

Edit6: TEdit;

Label7: TLabel;

Edit7: TEdit;

Label8: TLabel;

Button1: TButton;

Chart1: TChart;

Series1: TPointSeries;

Series2: TLineSeries;

Series3: TLineSeries;

Series4: TLineSeries;

procedure Button1Click(Sender: TObject);

procedure Edit1KeyPress(Sender: TObject; var Key: Char);

procedure Edit2KeyPress(Sender: TObject; var Key: Char);

procedure Edit3KeyPress(Sender: TObject; var Key: Char);

procedure Edit4KeyPress(Sender: TObject; var Key: Char);

procedure Edit5KeyPress(Sender: TObject; var Key: Char);

procedure Edit6KeyPress(Sender: TObject; var Key: Char);

procedure Edit7KeyPress(Sender: TObject; var Key: Char);

private

procedure KeyPress(var Key: Char);

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

x0,

x0z,

xk,

x,

x1,

x2,

U,

h,

hz,

eps,

K: Extended;

n: Byte;

y,

yz,

y0,

y0z,

f: Array [1..5] of double;

implementation

{$R \*.dfm}

procedure equation(x: real);

begin

if y[1] <= 0 then // Формирование управления

k:=1

else

k:=-1;

if (k \* sqrt( abs(2 \* y[1]) ) - y[2]) >= 0 then

u:=1

else

u:=-1;

f[1] := y[2]; // Уравнения движения объекта

f[2] := u;

end;

procedure Euler(x0: real); // Процедура обращения к методу Эйлера 1-го порядка

var

x: Double;

i: Byte;

begin

equation(x0);

for i := 1 to n do

y[i] := y0[i] + h \* f[i];

end;

procedure TForm1.Button1Click(Sender: TObject);

var

i,

j: Integer;

begin

case Button1.Tag of

0 : begin

Button1.Tag := 1;

Button1Click(Sender);

end;

1 : begin

Button1.Tag := 0;

end;

end;

//================= Обозначение начальных условий ==============================

n := StrToInt(Edit3.Text); // Порядок системы уравнений

x0 := StrToInt(Edit4.Text); // Начальное время процесса

xk := StrToInt(Edit5.Text); // Конечное время процесса

h := StrToFloat(Edit6.Text); // Начальный шаг интегрирования

eps := StrToFloat(Edit7.Text); // Относительная погрешность

y0[1] := StrToFloat(Edit1.Text); // Начальные координаты объекта - X1

y0[2] := StrToFloat(Edit2.Text); // Начальные координаты объекта - X2

//==============================================================================

hz:=h; // Помещение начального шага интегрирования в буфер

x0z:=x0; // Помещение начального времени процесса в буфер

x:=x0;

while (sqr( y0[1]) + sqr(y0[2]) ) > 0.001 do // Условие прекращения счета

begin

Euler(x0); // 1-й шаг на величину h от начальной точки

for i := 1 to N do

yz[i] := y[i]; // Запоминание значений Y[i]

h := h / 2; // Уменьшение величины шага в двое

Euler(x0); // 2-й шаг на величину h/2 от начальной точки

for i := 1 to N do

y0[i] := y[i]; // формирование новых начальных условий для Y0[i]

x0 := x0 + h; // формирование новых начальных условий для x0

Euler(x0); // 3-й шаг на величину h/2 от середины отрезка

j := 0; // Точность определяется когда счётчик достигнет N

for i := 1 to n do // Проверка обеспечения точности по всем переменным

if abs( yz[i] - y[i] ) < (eps \* abs(yz[i])) then

inc(j);

if j = n then // Условия достижения точности

begin

x0 := x0 + h; // Формирование новых начальных условий

x0z := x0; // Запоминание новых начальных условий

for i:=1 to n do

begin

y0[i] := y[i];

y0z[i] := y[i];

end;

h := hz; // Восстановление исходного значение шага интегрирования

Chart1.Series[0].AddXY(y[1],y[2]); // Вывод результатов решения на график

Chart1.Series[1].AddXY(0,y0[2]); // Ось оринат

Chart1.Series[1].AddXY(0,-y0[2]);

Chart1.Series[2].AddXY(-y0[1],0); // Ось абсцисс

Chart1.Series[2].AddXY(y0[1],0);

//Chart1.Series[1].AddXY(x0,y[1]);

//Chart1.Series[2].AddXY(x0,y[2]);

Chart1.Refresh; // Обновление графика

end

else // В случае, если точность не достигнута происходит возврат в исходную точку с исходными значениями

begin

x0 := x0z;

x := x0;

for i := 1 to n do

y0[i] := y0z[i];

end;

end;

Chart1.SaveToBitmapFile('Phase\_Trajectories.bmp'); // Сохранение изображения

end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char); ); // Вызов защиты от ввода не цифр

begin

KeyPress(Key);

end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);

begin

KeyPress(Key);

end;

procedure TForm1.Edit3KeyPress(Sender: TObject; var Key: Char);

begin

KeyPress(Key);

end;

procedure TForm1.Edit4KeyPress(Sender: TObject; var Key: Char);

begin

KeyPress(Key);

end;

procedure TForm1.Edit5KeyPress(Sender: TObject; var Key: Char);

begin

KeyPress(Key);

end;

procedure TForm1.Edit6KeyPress(Sender: TObject; var Key: Char);

begin

KeyPress(Key);

end;

procedure TForm1.Edit7KeyPress(Sender: TObject; var Key: Char);

begin

KeyPress(Key);

end;

procedure TForm1.KeyPress(var Key: Char); // Защита от ввода не цифр

begin

if not (Key in ['0'..'9',',',#8,'-']) then

Key := #0;

end;

end.

1.4 Используемые технические средства

Для работы программы рекомендуется использовать электронно-вычислительные машины со следующей минимальной конфигурацией:

* процессор – частота не ниже 500 МГц;
* ОЗУ – не менее 256 МБ;
* жесткий диск объёмом – 16 ГБ;
* монитор – разрешение не менее 1024x768 точек;
* клавиатура;
* мышь.

1.5 Вызов и загрузка

Запуск программы производится по двойному нажатию на файл «Differential\_Equations.exe», который изображен на рисунке 7.

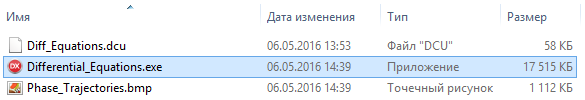


Рисунок 7 – Файл «WorkingWithArrays.exe»

Программа имеет объём 17,5Мб на диске.

Для функционирования программы из оперативной памяти под неё выделяется приблизительно 8 Мб.

Окно программы с контрольным примером ввода данных представлено на рисунке 8.

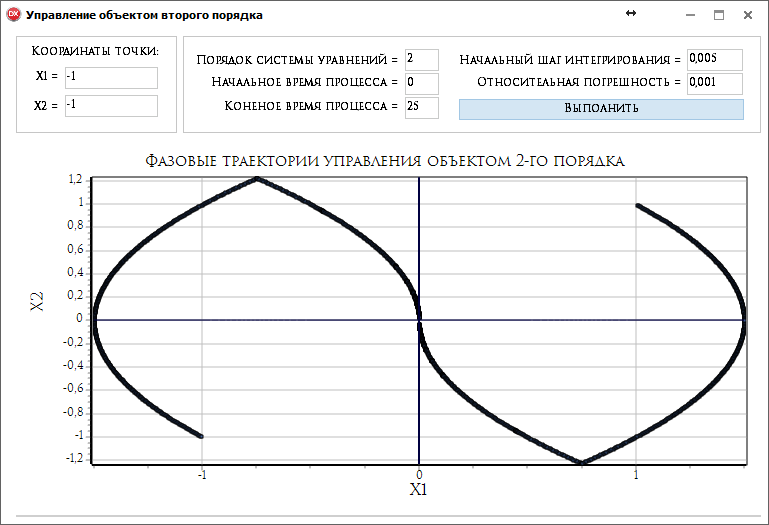


Рисунок 8 – Окно программы с примером ввода данных

1.6 Входные данные

Для работы программы следующие данные являются входными:

* координаты точки X1;
* координаты точки X2;
* порядок системы уравнений;
* начальное время процесса;
* конечное время процесса;
* начальный шаг интегрирования;
* относительная погрешность.

1.7 Выходные данные

Выходными данными работы программы являются следующие данные:

* значения X1 и X2 для построения фазовых траекторий управления объектом второго порядка;
* график фазовых траекторий управления объектом второго порядка.

# Заключение

В результате выполнения лабораторной работы был изучен процесс работы с определением фазовых траекторий при оптимальном по быстродействию управлении объектом второго порядка опираясь на использование методов решения дифференциальных уравнений и полученные теоретические знания были применены на практике.

В результате работы с массивами был получен график фазовых траекторий при оптимальном по быстродействию управлении объектом второго порядка в виде графического изображения формата «.bmp».