Министерство образования и науки РФ
ФГБПОУ ВПО Тульский государственный университитет
КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

ОДНОШАГОВЫЕ МЕТОДЫ РЕШЕНИЯ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Лабораторная работа № 6
по курсу «Вычислительный практикум»

Вариант № 4

Выполнил: студент группы 220601 ________ Белым А.А.
(подпись)
Проверил: к. т. н., доцент ________ Карцева А.С.
(подпись)

Тула 2011

Цель работы

Цель работы заключается в том, чтобы изучить различные методы решения дифференциальных уравнений и написать программу, реализующий один из таких методов.

Задание на работу

Решить дифференциальное уравнение методом Эйлера

$$y' = \frac{k}{x^2} - py^2$$

с заданными начальными условиями $y(a) = g$ на отрезке $[a, b]$ с шагом h.

Вариант задания для тестового примера:

$$k = 0.4,\ p = 0.4;\ a = 1,\ b = 2,\ g = 1;\ h = 0.1$$

Теоретическая справка

Дифференциальные уравнения связаны с построением моделей динамики (движения) объектов исследования. Они описывают, как правило, изменение параметров объектов во времени (хотя могут быть и другие случаи). Результатом решения дифференциальных уравнений являются функции, а не числа, как при решении алгебраических уравнений, поэтому они и более трудоемки.

При использовании численных методов решение дифференциальных уравнений представляется в табличном виде, т.е. получается совокупность значений (X_n, Y_n). Решение носит шаговый характер, т.е. по одной или нескольким начальным точкам (X, Y) за один шаг находят следующую точку, затем следующую и т.д. Решение между двумя соседними значениями аргумента $h = X_{n+1} - X_n$ называется шагом.

Однако прежде чем обсуждать методы решения, приведем некоторые сведения из курса дифференциальных уравнений.

В зависимости от числа независимых переменных, дифференциальные уравнения делятся на две категории: обыкновенные дифференциальные уравнения (ОДУ), содержащие одну независимую переменную, и уравнения с частными производными, содержащими несколько независимых переменных (например, в механике сплошных сред искомой функцией является плотность, t^0 , напряжение и др., а аргументами - координаты рассматриваемой точки в пространстве и время).

Обыкновенные дифференциальные уравнения могут содержать одну или несколько производных от искомой функции $y = f(x)$ и могут быть записаны в виде:

$$F(x, y, y', ..., y^{(n)}) = 0, \tag{1}$$

где x – независимая переменная.

Наивысший порядок (n) производной, входящей в уравнение (1) называется порядком дифференциального уравнения. В частности
$F(x, y, y') = 0$ - дифференциальное уравнение I порядка.
$F(x, y, y', y'') = 0$ - дифференциальное уравнение II порядка.
В ряде случаев удается выразить старшую производную в явном виде

$$y' = f(x, y); y'' = f(x, y, y').$$

Такие уравнения называют уравнениями, разрешенными относительно старшей производной.

Линейным дифференциальным уравнением называется уравнение, линейное относительно искомой функции и её производных.

Решением дифференциального уравнения (1) n -го порядка называется всякая функция $y = \phi(x)$, которая после ее подстановки в (1) превращает его в тождество. Решение ОДУ может быть общим и частным.

Общее решение ОДУ n -го порядка содержит n произвольных постоянных $C_1, C_2, C_3, ..., C_n$, т.е. решение ОДУ имеет вид: $y = \phi(x, C_1, C_2, ..., C_n)$.

Частное решение ОДУ получается из общего, если произвольным постоянным задать определенные значения.

Будем искать решение на ряде дискретных точек $t_0, t_1, ..., t_n$, удаленных друг от друга на расстоянии $h = t_{n+1} - t_n = const$, в виде

$$x(t_1) = x(t_0) + \int_{t_0}^{t_1} f(x, t)dt,$$

полученном путем интегрирования уравнения dx=f(x,t)dt .

Если принять, что на отрезке $[t0, t1] x' = x'(t_0) = f(x_0, t_0) = const$, то

$$x(t_1) = x(t_0) + f(x, t)(t_1 - t_0)$$

или, обозначив $t_1 - t_0 = h$, в дискретном виде

$$x_1 = x_0 + x_0' h.$$

Для точки x_{n+1} можно записать

$$x_{n+1} = x_n + x_n' h.$$

Полученное выражение известно как явный (прямой) метод Эйлера.

Искомая функция x(t) на шаге интегрирования была аппроксимирована прямой, совпадающей с касательной в точке $x_n = x(t_n)$.

В указанном выражении производная вычислялась в точке (x_0, t_0). Можно также выразить x_1 через x_0 и производную в точке (x_1, t_1), т.е. $x_1' = f(x_1, t_1)$. Тогда получим

$$x_1 = x_0 + x_1' h$$

Или в общем виде

$$x_{n+1} = x_n + x_{n+1}' h$$

Эта формула называется неявным (обратным) методом Эйлера .

Последнюю формулу можно представить в виде $x_{n+1} = x_n + f(x_{n+1}, t_{n+1})h$, где x_{n+1} входит и в правую часть . Поэтому эта формула пригодна, когда будет предсказано значение x_{n+1}, например, с помощью явного метода Эйлера. Таким образом, мы пришли к понятию «предсказание», когда определяется значение искомой функции в последующей точке. На основе найденного «предсказания» можно рассчитать значение $x_{n+1}' = f(x_{n+1}, t_{n+1})$ и использовать его при коррекции, которую выполним по неявной формуле Эйлера. Из-за ошибки «предсказания» может быть получена неточная коррекция. Чаще всего «предсказание» используется в качестве начального приближения для решения уравнения методом Ньютона.

Еще одну формулу численного интегрирования можно получить, приняв $f(x, t) = \frac{1}{2}(x_n' + x_{n+1}')$, тогда:

$$x_{n+1} = x_n + \frac{1}{2}(x_n' + x_{n+1}').$$

Это формула трапеции, которую иногда называют модифицированным методом Эйлера.

Это также неявная формула интегрирования, т. к. неизвестная величина x'_{n+1} входит в правую часть. Значение переменной x_{n+1} получают из решения нелинейного алгебраического уравнения

$$x_{n+1} = x_n + \frac{1}{2}(f(x_n, t_n) + f(x_{n+1}, t_{n+1})$$

методом Ньютона.

Алгоритм неявного метода Эйлера отличается от алгоритма метода трапеции отсутствием в формуле определения x составляющей $f(x_0, t_0)$ и вместо $\frac{1}{2}h$ используется h .

Схема алгоритма

На рисунке 1 представлена схема алгоритма расчета узлов и получения решения дифференциального уравнения по явному методу Эйлера.

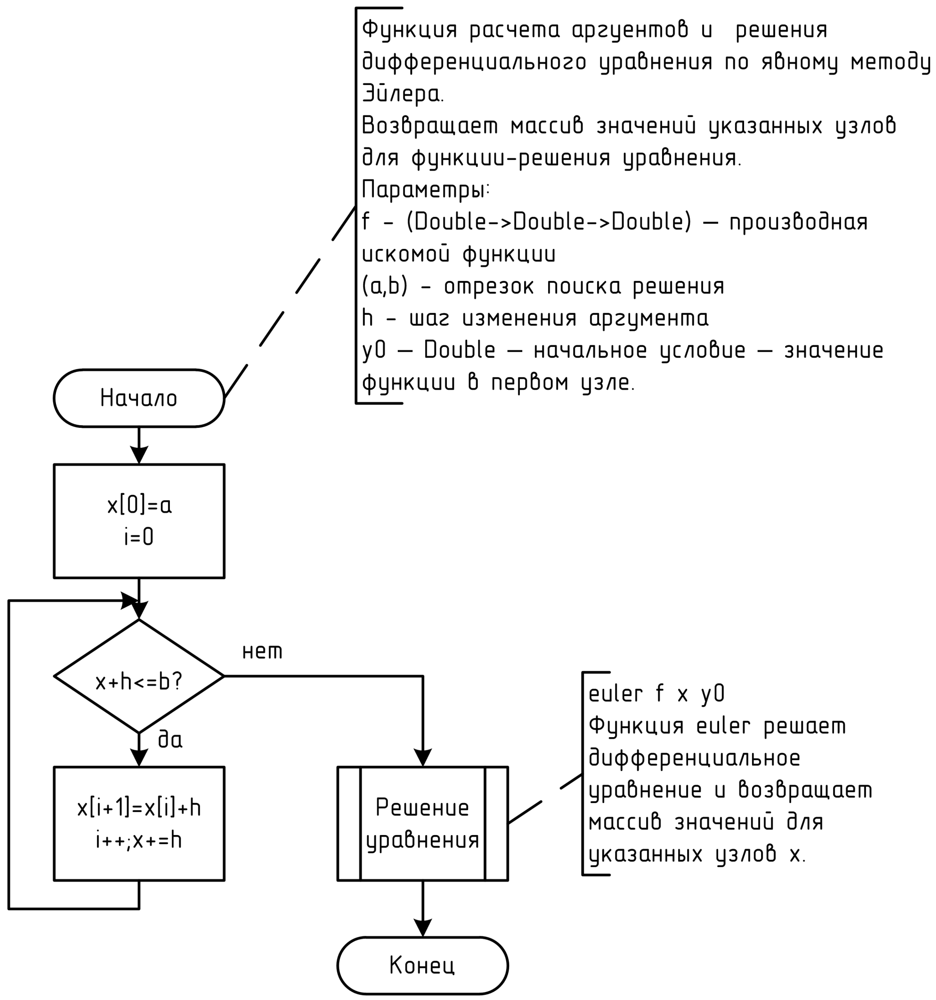

Рисунок 1 - Схема алгоритма расчета узлов и получения решения уравнения

На рисунке 2 представлена схема алгоритма решения дифференциального уравнения по явному методу Эйлера.

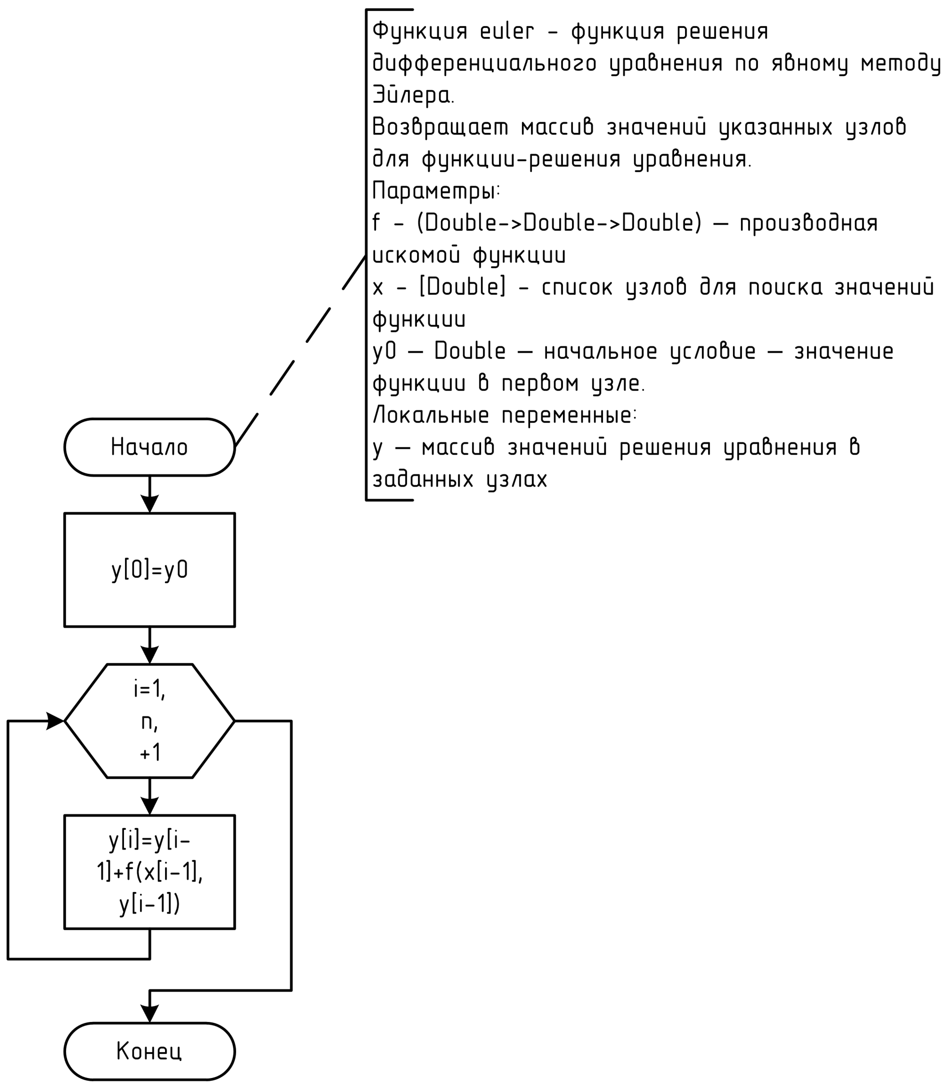

Рисунок 2 - Схема алгоритма решения уравнения по явному методу Эйлера

Инструкция пользователя

Программа позволяет решенить дифференциальное уравнение с пмощью явного метода Эйлера.

Программе необходимо передать границы диапазона поиска решения, шаг изменения аргумента в этом диапазоне, начальное условия - значение функции в нижней границе диапазона и параметры производной функции. Все данные вводятся в специально отведенные поля. Росле завершения ввода нажмите клавишу "Посчитать".

После завершения расчетов программа выведет на экран значение решения уравнения в заданных узлах.

Инструкция программиста

При разработке программы решения дифференциального уравнения по явному методу Эйлера были написаны следующие функции:

1. Функция euler - функция решения дифференциального уравнения по явному методу Эйлера.

Возвращает массив значений указанных узлов для функции-решения уравнения.

euler::(Double->Double->Double)->[Double]->Double->[Double]

Параметры функции представлены в таблице 1 :

Таблица 1 - Параметры функции решения дифференциального уравнения

имя	**тип**	**предназначение**
f	(Double->Double->Double)	производная искомой функции
x	[Double]	список узлов для поиска значений функции
y0	Double	начальное условие – значение функции в первом узле.

Локальные переменные функции представлены в таблице 2 :

Таблица 2 - Локальные переменные функции решения дифференциального уравнения

имя	тип	предназначение
euler’	[Double]	список значений решения уравнения в заданных узлах
derivat	([Double],[Double])	кортеж списка значений производной в узлах и шага между соседними узлами.
deltas	[Double]	список шагов между соседними узлами.
derivates	[Double]	список значений производной в узлах.

Текст программы

Реализиция задачи решения дифференциального уравнения по явному методу Эйлера написана на языке Haskell 98 и состоит из двух частей.

Первая часть, файл Euler.hs, содержит вычислительное ядро. Исходный текст этого модуля приводится ниже.

```
module Euler where
f' k p x y=k/x^2-p*y^2::Double
euler::(Double->Double->Double)->[Double]->Double->[Double]
-- Функция euler - функция решения дифференциального уравнения
--    по явному методу Эйлера.
-- Возвращает массив значений указанных узлов для функции-решения уравнения.
-- Параметры:
-- f - (Double->Double->Double) - производная искомой функции
-- x - [Double] - список узлов для поиска значений функции
-- y0 - Double - начальное условие - значение функции в первом узле.
-- Локальные переменные:
-- euler' - [Double] - список значений решения уравнения в заданных узлах
-- derivat - ([Double],[Double]) - кортеж списка значений производной в узлах
    -- и шага между соседними узлами.
-- deltas - [Double] - список шагов между соседними узлами.
-- derivates - [Double] - список значений производной в узлах.
euler f' xs y0=euler' where
    euler'=y0:zipWith getNewValue euler' derivat
        where
            getNewValue y (y',h)=y+y'*h
            derivat = zip (derivatives xs) (deltas xs)
            deltas []=[]
            deltas (x:[])=[]
            deltas (x0:x1:xs)=x1-x0:deltas(x1:xs)
            derivatives xs=zipWith f' xs euler'
```

Вторая часть - файл Inter.hs - является графическим интерфейсом для вычислительного ядра первого модуля. Далее представлен текст этого второго модуля.

```
module Main where
import Prelude hiding (catch)
import Control.Exception
import Graphics.UI.Gtk
import Graphics.UI.Gtk.Builder
import Euler


getValue:: Entry->IO (Double)
getValue entry=do
    y<-get entry entryText
    evaluate (read y)

onClickExit=mainQuit

getValues::[Entry]->IO [Double]
getValues xs=getValues' xs []

getValues'::[Entry]->String->IO [Double]
getValues' [] ers |ers==[]= return []
                  |otherwise= error ers

getValues' (x:xs) ers=do
    y<-try(getValue x)::IO(Either SomeException Double)
```

```haskell
     case y of
         Left e->do
             v<-entryGetText x
             vs<-getValues' xs (ers++"Неправильное значение: \"" ++ v++"\"!\n")
             return vs
         Right v->do
             vs<-getValues' xs ers
             return (v:vs)
main::IO ()
main = do
   let

   initGUI
   gtkbuilder<-builderNew
   builderAddFromFile gtkbuilder "gui.glade"
   let
        showFail::SomeException-> IO()
        showFail e=do
            dlg<-messageDialogNew Nothing [DialogModal] MessageError
                ButtonsClose ("Ошибки:\n"++ show e)
            _<-dialogRun dlg
            widgetDestroy dlg
        showRes _ [] =[];showRes [] _=[]
        showRes (x:xs) (y:ys)="f("++(show x)++")="++(show y)++"\n"
                                    ++(showRes xs ys)
        onClickRun=do
            entryA <- builderGetObject gtkbuilder castToEntry "entry1"
            entryB <- builderGetObject gtkbuilder castToEntry "entry2"
            entryH <- builderGetObject gtkbuilder castToEntry "entry3"
            entryY0 <- builderGetObject gtkbuilder castToEntry "entry4"
            entryK <- builderGetObject gtkbuilder castToEntry "entry5"
            entryP <- builderGetObject gtkbuilder castToEntry "entry6"
            labelResult <- builderGetObject gtkbuilder castToLabel "labelResult"
            [a,b,h,y0,k,p]<-getValues [entryA,entryB,entryH,entryY0,entryK,entryP]
            if a>=b then error "Верхняя граница меньше/равна нижней!" else
                if h<=0 then error "Шаг меньше/равен 0!" else do
                        let
                            xs=[a,a+h..b]
                            res=euler (f' k p) xs y0
                        set labelResult [labelText:="Результат:\n"++
                                              (showRes xs res)]
   window    <- builderGetObject gtkbuilder castToWindow "window1"
   buttonRun <- builderGetObject gtkbuilder castToButton "buttonRun"
   onClicked buttonRun (catch onClickRun showFail)
   buttonExit <- builderGetObject gtkbuilder castToButton "button2"
   onClicked buttonExit onClickExit
   onDestroy window mainQuit
   widgetShowAll window
   mainGUI
```

Тестовый пример

Ниже на рисунке 3 представлен пример работы программы при решении дифференциального уравнения $y' = \frac{k}{x^2} - py^2$ с параметрами $k = 0.4, p = 0.4$ на отрезке $[a, b] = [1, 2]$ с шагом $h = 0.1$ и начальным условием $y(1) = 1$ по явному методу Эйлера.

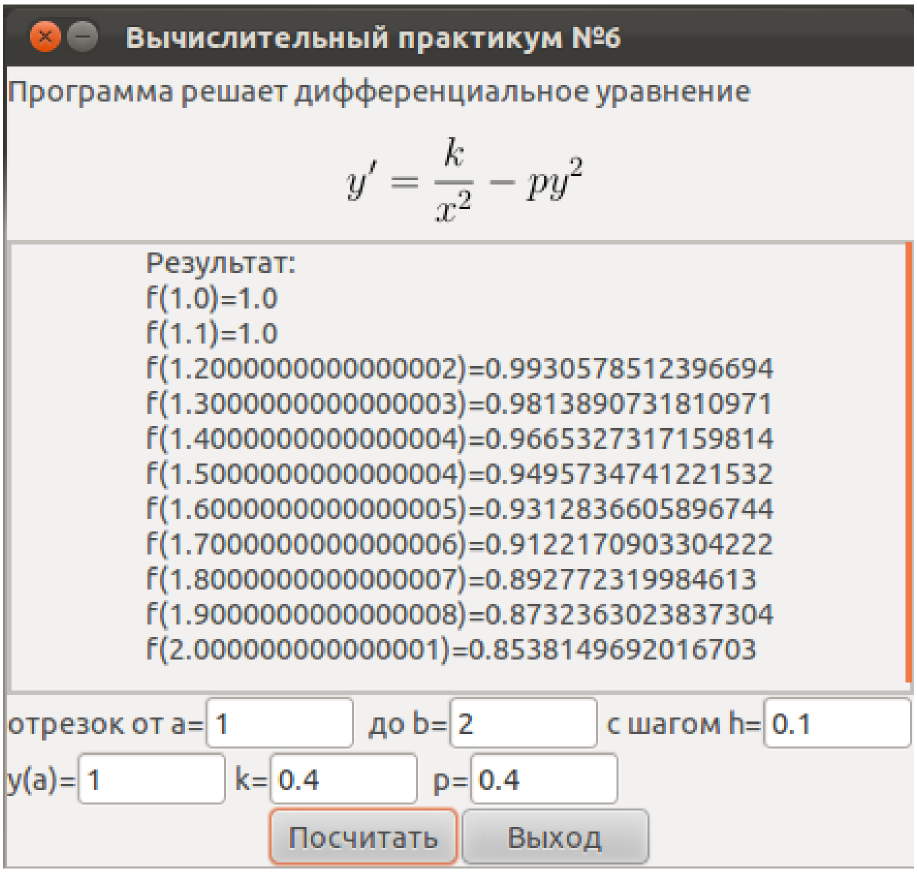

Рисунок 3 - Пример работы программы решения дифференциального уравнения

Вывод

В этой лабораторной работе я изучил различные одношаговые методы решения дифференциальных уравнений. Решение дифференциальных уравнений представляет важную задачу, так как широко применяется в различных областях науки и техники. Ручное решение может быть слишком долгим и трудоёмким, поэтому необходимо уметь применять численные методы решения дифференциальных уравнений с использованием вычислительной техники.