Содержание

Введение 3

1. Постановка задачи 5
2. Обзор существующих методов решения задачи 6 2.1.Метод Рунге-Кутта четвертого порядка для решения

уравнения первого порядка 6

2.2.Задача Коши 6

2.3.Метод Булирша- Штера с использованием

рациональной экстраполяции для системы уравнений 7

2.4 Метод Адамса 8

2.5. Метод Эйлера 9

3. Описание алгоритмов решения задания 13

3.1. Описание переменных 13

3.2. Блок- схема главного модуля 14

3.3. Описание алгоритма главной программы 14

3.4. Блок-схема функции “func” 15

3.5. Описание блок- схемы функции “func” 15

4. Описание программного обеспечения 16

4.1. Описание операционной системы 16

4.2. Описание языка программирования 18

4.3. Описание программы 19

5. Контрольный пример 21

6.Анализ полученных результатов 22

Список литературы 24

Приложение 25

Введение

Уравнение  называется *обыкновенным дифференциальным n-го порядка,* если F определена и непрерывна в некоторой области и, во всяком случае, зависит от . Его решением является любая функция *u(x),* которая этому уравнению удовлетворяет при всех *x* в определённом конечном или бесконечном интервале. Дифференциальное уравнение, разрешенное относительно старшей производной имеет вид



Решением этого уравнения на интервале I=[a,b] называется функция *u(x)*

Решить дифференциальное уравнение у/=f(x,y) численным методом - это значит для заданной последовательности аргументов х0, х1…, хn и числа у0, не определяя функцию у=F(x), найти такие значения у1, у2,…, уn, что уi=F(xi)(i=1,2,…, n) и F(x0)=y0.

Таким образом, численные методы позволяют вместо нахождения функции y=F(x) (3) получить таблицу значений этой функции для заданной последовательности аргументов. Величина h=xk-xk-1  называется шагом интегрирования.

Метод Эйлера относиться к численным методам, дающим решение в виде таблицы приближенных значений искомой функции у(х). Он является сравнительно грубым и применяется в основном для ориентировочных расчетов. Однако идеи, положенные в основу метода Эйлера, являются исходными для ряда других методов.

Метод Эйлера для обыкновенных дифференциальных уравнений используется для решений многих задач естествознания в качестве математической модели. Например задачи электродинамики системы взаимодействующих тел (в модели материальных точек), задачи химической кинетики, электрических цепей. Ряд важных уравнений в частных производных в случаях, допускающих разделение переменных, приводит к задачам для обыкновенных дифференциальных уравнений – это, как правило, краевые задачи (задачи о собственных колебаниях упругих балок и пластин, определение спектра собственных значений энергии частицы в сферически-симметричных полях и многое другое).

1.Постановка задачи

1.1. Решить приближенно дифференциальное уравнение вида методом Эйлера

1.2. Составить блок-схему алгоритма для решения данного задания.

1.3. Разработать программу на языке Microsoft Visual C++

1.4. Протестировать программу на примере y’=2x+y (n=5, [0,1], y0=1)

1.5. Выполнить анализ результатов.

1.6. Оформить пояснительную записку с приложением.

2.Обзор методов решения задачи.

2.1. *Метод Рунге-Кутта четвертого порядка для решения уравнения первого порядка.*

Идея Рунге-Кута состоит в том, чтобы использовать метод неопределённых коэффициентов. Наиболее употребительным методом Рунге-Кутта решения уравнения первого порядка y' = F(x,y) (2.1.1) является метод четвертого порядка, в котором вычисления производятся по формуле:

yk+1 = yk +(k1 +2k2 +2k3 +k4 )/6, (2.1.2)

*где*

k1 = Fk h = F(xk , yk )h

k2 = F(xk +h/2, yk +k1 /2)h

k3 = F(xk +h/2, yk +k2 /2)h

k4 = F(xk +h, yk +k3 )h,

k = 0, ..., n-1

h = (xf -x0 )/n (2.1.3)

2.2. *Задача Коши.*

Рассмотрим задачу Коши для уравнений первого порядка на отрезке [a,b]:

,  (2.1.4)

Разобьём промежуток [a,b] на N частей . Обозначим , где u(x) –точное решение задачи Коши, и через  значения приближенного решения в точках . Существует 2 типа численных схем :

1. явные: ) (2.2.1)
2. неявные:  (2.2.2)

Здесь F некоторая функция, связывающая приближения. В явных схемах приближенное значение  в точке  определяется через некоторое число k уже определённых приближенных значений. В неявных схемах  определяется не рекурентным способом, как в явных схемах, а для его определения возникает уравнение, поскольку равенство (2.2.2) представляет из себя именно уравнение на . Явные схемы проще, однако зачастую неявные схемы предпочтительнее.

2.3. *Метод Булирша-Штера с использованием рациональной экстраполяции для системы уравнений*

Метод Булирша-Штера (Bulirsch-Stoer Method) - это метод решения системы обыкновенных дифференциальных уравнений первого порядка с гладкими правыми частями. Гладкость правых частей является необходимой для работы метода. Если правые части вашей системы не являются гладкими или содержат разрывы, то лучше использовать метод Рунге-Кутта. В случае же гладкой системы метод Булирша-Штера позволяет добиться существенно большей точности, чем метод Рунге-Кутта.

*Принцип работы метода*

Основной идеей метода является вычисление состояния системы в точке x+h, как результата двух шагов длины h/2, четырех шагов длины h/4, восьми шагов длины h/8 и так далее с последующей экстраполяцией результатов. Метод строит рациональную интерполирующую функцию, которая в точке h/2 проходит через состояние системы после двух таких шагов, в точке h/4 проходит через состояние системы после четырех таких шагов, и т.д., а затем вычисляет значение этой функции в точке h = 0, проводя экстраполяцию.

Гладкость правых частей приводит к тому, что вычисленное при помощи экстраполяции состояние системы оказывается очень близко к действительному, а использование рациональной экстраполяции вместо полиномиальной позволяет ещё больше повысить точность.

Таким образом проводится один шаг метода, после чего принимается решение - следует ли изменять шаг, а если да - то в какую сторону. При этом используется оценка погрешности, которую мы получаем в качестве дополнительного результата при рациональной экстраполяции. Следует отметить, что алгоритм решает автономную систему, т.е. если уравнения системы содержат время, то необходимо ввести время в качестве переменной, производная от которой тождественно равна единице.

* 1. *Метод Адамса*

*Явная схема Адамса.*

Рассмотренные выше методы являются явными одношаговыми (для нахождения последующего приближения используется лишь одно предыдущее). Приведённый ниже метод является многошаговым.

Пусть задана задача Коши:

 (2.4.1)

Для точного решения (которое нам не известно) выполнено:

 (2.4.2)

Предположим, нам известны приближенные значения  функции *u(x)* в *k* точках  (стартовые *k* точек, в частности, можно найти методом Эйлера или методом Рунге-Кутта того или иного порядка), тогда функцию *f(x,u(x))* в (2.4.2) для приближенного вычисления интеграла можно заменить на интерполяционный полином порядка *k-1,* построенный по *k* точкам , интеграл от которого считается явно и представляет собой линейную комбинацию значений c некоторыми множителями . Таким образом, мы получаем следующую рекуррентную процедуру вычисления приближенных значений  функции *u(x)* (являющимся точным решением задачи Коши) в точках  :

 (2.4.3)

Описанная схема является *k-*шаговой явной формулой Адамса.

*Неявная схема Адамса.*

Пусть - интерполяционный полином порядка *k*, построенный по *k+1* значению б одно из которых, именно , мы будем считать неизвестным. Модифицируем (2.4.3), заменив в нём  на полином более высокой степени , интеграл от которого выражается в виде линейной комбинации значений  с некоторыми новыми коэффициентами :

 (2.4.4)

Формула (2.4.4) представляет собой неявную схему Адамса и является уравнением на , которое можно решать методом последовательных приближений. Естественно, что начальное приближение , должно быть разумно выбрано. Для этого удобно объединить явную и неявную схемы Адамса в одну, называемую «методом коррекции». Именно с помощью явной схемы определяется начальное приближение (прогноз), а затем по неявной схеме оно необходимое число раз (обычно один или два) корректируется методом последовательных приближений до достижения заданной точности (коррекция).

2.5.*Метод Эйлера****.***

Решить дифференциальное уравнение у/=f(x,y) численным методом - это значит для заданной последовательности аргументов х0, х1…, хn и числа у0, не определяя функцию у=F(x), найти такие значения у1, у2,…, уn, что уi=F(xi)(i=1,2,…, n) и F(x0)=y0. (2.5.1)

Таким образом, численные методы позволяют вместо нахождения функции

У=F(x) получить таблицу значений этой функции для заданной последовательности аргументов. Величина h=xk-xk-1 называется ***шагом интегрирования.***

Метод Эйлера относиться к численным методам, дающим решение в виде таблицы приближенных значений искомой функции у(х). Он является сравнительно грубым и применяется в основном для ориентировочных расчетов. Однако идеи, положенные в основу метода Эйлера, являются исходными для ряда других методов.

Рассмотрим дифференциальное уравнение первого порядка (2.5.1)

с начальным условием

x=x0, y(x0)=y0  (2.5.2)

Требуется найти решение уравнения (2.5.1) на отрезке [а,b].

Разобьем отрезок [a, b] на n равных частей и получим последовательность х0, х1, х2,…, хn, где xi=x0+ih (i=0,1,…, n), а h=(b-a)/n-шаг интегрирования.

В методе Эйлера приближенные значения у(хi)≈yi вычисляются последовательно по формулам уi+hf(xi, yi) (i=0,1,2…).

При этом искомая интегральная кривая у=у(х), проходящая через точку М0(х0, у0), заменяется ломаной М0М1М2… с вершинами Мi(xi, yi) (i=0,1,2,…); каждое звено МiMi+1 этой ломаной, называемой ***ломаной Эйлера,*** имеет направление, совпадающее с направлением той интегральной кривой уравнения (2.5.1), которая проходит через точку Мi. Если правая часть уравнения (2.5.1) в некотором прямоугольнике R{|x-x0|≤a, |y-y0|≤b}удовлетворяет условиям:   
 |f(x, y1)- f(x, y2)| ≤ N|y1-y2| (N=const), (2.5.3)

|df/dx|=|df/dx+f(df/dy)| ≤ M (M=const),

то имеет место следующая оценка погрешности:

|y(xn)-yn| ≤ hM/2N[(1+hN)n-1], (2.5.4)

где у(хn)-значение точного решения уравнения (2.5.1) при х=хn, а уn- приближенное значение, полученное на n-ом шаге.

Формула (13) имеет в основном теоретическое применение. На практике иногда оказывается более удобным *двойной просчет*: сначала расчет ведется с шагом h, затем шаг дробят и повторный расчет ведется с шагом h/2. Погрешность более точного значения уn\* оценивается формулой

|yn-y(xn)|≈|yn\*-yn|. (2.5.5)

Метод Эйлера легко распространяется на системы дифференциальных уравнений и на дифференциальные уравнения высших порядков. Последние должны быть предварительно приведены к системе дифференциальных уравнений первого порядка.

***Модифицированный метод Эйлера***

Рассмотрим дифференциальное уравнение (2.5.1) y/=f(x,y) с начальным условием y(x0)=y0. Разобьем наш участок интегрирования на n равных частей. На малом участинтегральную кривую заменим прямой линией.

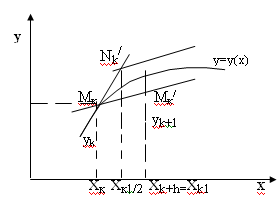


Рис.1 Метод Эйлера в графическом видa

Получаем точку Мк(хк,ук). Через Мк проводим касательную: у=ук=f(xk,yk)(x-xk). Делим отрезок (хк,хк1) пополам:

xNk/=xk+h/2=xk+1/2 (2.5.6)

yNk/=yk+f(xk,yk)h/2=yk+yk+1/2

Получаем точку Nk/. В этой точке строим следующую касательную:

y(xk+1/2)=f(xk+1/2, yk+1/2)=αk (2.5.7)

Из точки Мк проводим прямую с угловым коэффициентом αк и определяем точку пересечения этой прямой с прямой Хк1. Получаем точку Мк/. В качестве ук+1 принимаем ординату точки Мк/. Тогда:

ук+1=ук+αкh

xk+1=xk+h

(2.5.8) αk=f(xk+h/2, yk+f(xk,Yk)h/2)

yk=yk-1+f(xk-1,yk-1)h

(2.5.8)-рекурентные формулы метода Эйлера.

Сначала вычисляют вспомогательные значения искомой функции ук+1/2 в точках хк+1/2, затем находят значение правой части уравнения (11) в средней точке y/k+1/2=f(xk+1/2, yk+1/2) и определяют ук+1.

Для оценки погрешности в точке хк проводят вычисления ук с шагом h, затем с шагом 2h и берут 1/3 разницы этих значений:

| ук\*-у(хк)|=1/3(yk\*-yk), (2.5.9)

где у(х)-точное решение дифференциального уравнения.

Таким образом, методом Эйлера можно решать уравнения любых порядков. Например, чтобы решить уравнение второго порядка y//=f(y/,y,x) c начальными условиями y/(x0)=y/0, y(x0)=y0, выполняется замена:

y/=z (2.5.10)

z/=f(x,y,z)

Тем самым преобразуются начальные условия: y(x0)=y0, z(x0)=z0, z0=y/0. (2.5.12)

3.Описание алгоритмов решения задачи

3.1.Описание переменных.

|  |  |  |
| --- | --- | --- |
| ***Наименование*** | ***Тип*** | ***Описание*** |
| *Входные данные* | | |
| ***Xi*** | *double* | Начальное значение (x) интервала вычисления |
| ***Xkon*** | *double* | Конечное значение (x) интервала вычисления |
| ***n*** | *integer* | Количество шагов |
| ***Yi*** | *double* | Начальное значение y |
| ***kx*** | *double* | Коэффициент при переменой x |
| ***ky*** | *double* | Коэффициент при переменной y |
| *Выходные данные* | | |
| ***h*** | *double* | Фиксированное приращение аргумента (x) |
| ***res*** | *double* | Расчётное значение уравнение y’=F(x,y) в точке (x) |
| *Промежуточные* | | |
| ***i*** | *integer* | Счётчик цикла |
| ***Yprom*** | *double* | Промежуточное значение y в точке Xprom |
| ***Xprom*** | *double* | Промежуточное значение x при h/2 |
| ***a*** | *double* | Решение уравнения в точках f(Xprom,Yprom) |
| ***f1*** | *double* | Функция f(x,y) |

3.2. Блок- схема главного модуля

Начало

Xi,Xkon,n,Yi,kx,ky

i

1 n

1

func(Xi, Yi, kx, ky, h)

Конец

Res, Xi

h:=(Xi-Xkon)/n

3.3 Описание алгоритма главной программы**.**

|  |  |
| --- | --- |
| **Номер блока** | **Описание** |
| 1 | Ввод начального и конечного значений интервала вычисления уравнения, количество шагов, начальное значение у, а также коэффициенты при kx и ky. |
| 2 | Вычисление фиксированного приращения аргумента х |
| 3 | Цикл с шагом 1 и конечным значением не превышающим количество шагов, который высчитывает значение y на определённом интервале |
| 4 | Функция для расчёта уравнения вида y’=f(x,y); |
| 5 | Вывод результатов на интервале X |

3.4 Блок-схема функции “func”.

3.5 Описание блок- схемы функции “func”.

Начало

f1:=(kx\*Xi)+(ky\*Yi)

Yprom:=Yi+f1\*(h/2)

Xprom:=Xi+h/2;

A:=kx\*Xprom-Yprom

Yi:=Yi+a\*h;

Xi:=Xi+h

Xi,Yi

Конец

|  |  |
| --- | --- |
| **Номер**  **блока** | **Описание** |
| 1 | Вычисление: функции f1 с подстановкой начальных значений; промежуточных значений Yprom и Xprom, значения a для вычисления f(Xprom,Yprom) и расчёт результатов функции и переход на следующий шаг. |
| 2 | Приращение аргумента x на h |
| 3 | Вывод результатов уравнения и интервала |

\*Реализация алгоритма на языке программирования C++ представлена в приложении .

4.Описание программного обеспечения**.**

**4.1 Описание операционной системы**

Основное требование к операционной системе (ОС), предъявляемое поставленной задачей, это наличие ANSI или POSIX совместимого компилятора языка C++.

Для реализации задачи была выбрана последняя клиентская версия операционной системы Microsoft, основанная на ядре NT – Microsoft Windows XP Professional.

Указанная операционная система обладает рядом преимуществ:

* наличие достаточного количество ANSI или POSIX совместимых компиляторов языка C++, разработанных для данной ОС, а именно –
  + Microsoft C++ (version 2-6)
  + gcc
  + Borland C++
  + Intel C++
  + прочие;
* достаточная управляемость, надежность и безопасность;
* широкое распространение основанных на ядре NT операционных систем Microsoft, совместимых по программному обеспечению с Windows XP Professional (NT/2000/XP/2003 – client & server);
* высокая скорость работы приложений, разработанных для данной ОС с использованием компиляторов C++.

Исходный код программы может быть откомпилирован и под другой операционной системой, если для таковой имеется ANSI или POSIX совместимый компилятор языка C++.

Программа была протестирована на операционной системе Microsoft Windows XP Professional SP1.

Технические данные :

* HDD: 60 Gb
* Процессор x86 Family 15 Model 2 Stepping 7 GenuineIntel ~1817 МГц
* Версия BIOS Award Software International, Inc. F4, 06.03.2003
* Аппаратно-зависимый уровень (HAL) Версия = "5.1.2600.1106 (xpsp1.020828-1920)"
* Полный объем физической памяти 256,00 МБ
* Доступно физической памяти 29,97 МБ
* Всего виртуальной памяти 873,69 МБ
* Доступно виртуальной памяти 350,04 МБ
* Файл подкачки 618,21 МБ

4.2 Описание языка программирования

Язык программирования С++

С++ - это универсальный язык программирования, задуманный так, чтобы сделать программирование более приятным для серьезного программиста. За исключением второстепенных деталей С++ является надмножеством языка программирования C. Помимо возможностей, которые дает C, С++ предоставляет гибкие и эффективные средства определения новых типов. Используя определения новых типов, точно отвечающих концепциям приложения, программист может разделять разрабатываемую программу на лег ко поддающиеся контролю части. Такой метод построения программ часто называют абстракцией данных. Информация о типах содержится в некоторых объектах типов, определенных пользователем. Такие объекты просты и надежны в использовании в тех ситуациях, когда их тип нельзя установить на стадии компиляции. Программирование с применением таких объектов часто называют объектно-ориентированным. При правильном использовании этот метод дает более короткие, проще понимаемые и легче контролируемые программы.

В С++ нет типов данных высокого уровня и нет первичных операций высокого уровня. В нем нет, например, матричного типа с операцией обращения или типа строка с операцией конкатенации. Если пользователю понадобятся подобные типы, их можно определить в самом языке. По сути дела, основное, чем занимается программирование на С++ - это определение универсальных и специально-прикладных типов. Хорошо разработанный тип, определяемый пользователем, отличается от встроенного типа только способом определения, но не способом использования.

Реализация С++ очень легко переносима. Однако есть полные основания использовать С++ в среде, где имеется гораздо более существенная поддержка. Такие средства, как динамическая загрузка, пошаговая трансляция и база данных определений типов могут с пользой применяться без воздействия на язык.

Типы и средства сокрытия данных в С++ опираются на проводимый во время компиляции анализ программ с целью предотвращения случайного искажения данных. Они не обеспечивают секретности или защиты от умышленного нарушения правил. Однако эти средства можно использовать без ограничений, что не приводит к дополнительным расходам времени на выполнение или пространства памяти.

Компилятор Microsoft C++ и среда разработки Microsoft Visual Studio

В качестве компилятора для разработки приложения был выбран Microsoft C++ по следующим причинам:

* практически полная совместимость со стандартом ANSI C++;
* наличие удобной среды разработки Microsoft Visual Studio;
* наличие отличной документации;
* высокая скорость работы результирующих приложений;
* совместимость разработанных приложений с большим количеством широко распространенных операционных систем;
* достаточная скорость компиляции.

4.3 Описание программы

Разработанное приложение поставляется в виде 2-ух файлов:

1. method Eulera.cpp – исходный код программы на языке C++;
2. method Eulera.exe – исполняемый файл.

Для выполнения исполняемого файла необходима одна из ниже перечисленных операционных систем:

* Microsoft Windows 3.11+Win32s;
* Microsoft Windows 95/98/Me;
* Microsoft Windows NT/2000/XP/2003 – клиентская или серверная версия.

Программа не требует предварительной установки и может быть сразу же запущена на выполнение.

Исходный код приложения может быть откомпилирован в любом ANSI или POSIX совместимом компиляторе С++ для получения выполнимой программы. Для успешной компиляции требуется наличие стандартной библиотеки «iostream».

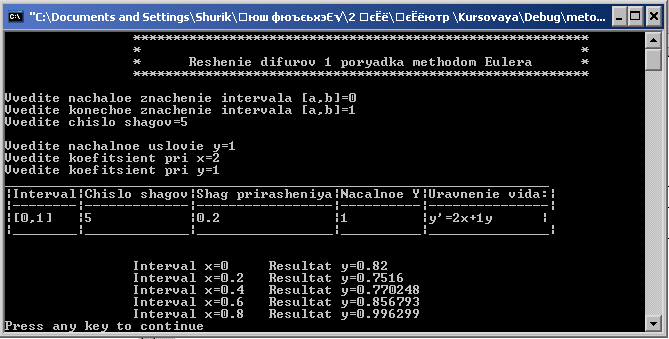
5. Контрольный пример

Данный метод протестирован на контрольном примере и реализован с помощью языка программирования С++.

В результате вычислений контрольного примера вида **y’=2x+y** с интервалом **[0,1],**

количеством шагов равному **5** и начальным условием **у** равным **1**, с помощью программы, получились следующие результаты:

Рис. 2. Экран с результатами выполнения программы.



Как видно, при вычислении программа на первом шаге берёт начальные значения для вычисления, а на последующих берёт значения полученные с предыдущих шагов. Можно сделать вывод, что точность вычисления данного метода зависит от количества выбранных шагов: чем больше шагов, тем меньше фиксированное приращение , а следовательно она более точно вычисляет значение всего интервала.

По работе программы стало видно, что с её использованием намного упростилась работа пользователя. Пользователь просто вводит интервал на котором должен вычисляться пример, количество шагов и начальное значения и программа выдаёт уже готовое решение данного примера.

6.Анализ полученных результатов.

По результатам программы можно составить таблицу сравнения результатов полученных при использовании программы и результатов, полученных ручным способом:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ручной способ вычисления** | | **Программный способ вычисления** | |
| Х | Y | X | Y |
| 0 | 0,82 | 0 | 0,82 |
| 0,2 | 0,75 | 0,2 | 0,7516 |
| 0,4 | 0,77 | 0,4 | 0,770248 |
| 0,6 | 0,85 | 0,6 | 0,856793 |
| 0,8 | 0,99 | 0,8 | 0,996299 |

Из приведенного сравнения можно сделать вывод, что один результат отличается от другого тем, что в примере, решенном программным способом ответ вычисляется с наибольшей точностью, чем при ручном способе. Это может быть связано с тем, что в ручном способе результат округляется для удобства вычисления примера.

Решение дифференциальных уравнений методом Эйлера можно также отобразить в графическом виде:



Рис.3.Графическое изображение решения примера y’=2x+y

Как видно из рис.3 графиком решения уравнения является кривая , форма которой зависит от количества разбиений интервала.

По результатам выполненной работы можно сделать вывод, что решение дифференциальных уравнений методом Эйлера является методом вычисления со средней точностью и точность вычисления данного метода зависит от количества разбиений интервала интегрирования. При сравнении результатов решенными разными способами можно сказать, что данный метод был верно реализован на языке программирования Microsoft Visual C++. Полученные результаты сходятся с небольшой погрешностью.

Список литературы.

1. Численные методы (анализ, алгебра, обыкновенные дифференциальные уравнения), Н.С. Бахвалов. Главная редакция физико- математической литературы изд-ва «Наука», М., 1975г.
2. Методы, теории обыкновенных дифференциальных уравнений. Н.И. Гаврилов . Государственное издательство «Высшая школа» Москва-1962г.
3. В.В.Пак., Ю.Л. Носенко. Высшая математика: Учебник.- Д.: Сталкер, 1997г.
4. Б. П. Демидович, И. А. Марон Основы вычислительной математике. – М., 1966
5. Загускин В. Л. – Справочник по численным методам решения уравнений. – М.: ФИЗМАТГИЗ, 1960. – 216 с.
6. Либерти, Джесс.

Освой самостоятельно С++ за 21 день, 4-е издание.:Пер с англ.-М.: Издательский дом «Вильямс», 2003.-832с.

1. П.Нортон, П.Иао «Программирование на С++ в среде Windows» («Диалектика» Киев 2003г.)
2. Янг М. Microsoft Visual C++ - М.:ЭНТРОП, 2000.
3. Марченко А.И., Марченко Л.А. – Программирование в среде

Turbo Pascal 7.0 – К.: ВЕК+, М.: Бином Универсал, 1998. – 496 с.

10. Высшая математика: Справ. материалы: Книга для учащихся .- М.:

Просвещение, 1988.-416 с.: ил.

Приложение.

Листинг программы.

#include<iostream>

using namespace std;

void func(double& Xi, double& Yi,double kx, double ky, double h);

int main()

{

double h,Xi,Yi,Xkon,kx,ky;

int n;

cout<<"\t"<<"\t"<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*n";

cout<<"\t"<<"\t"<<"\* \* "<<"\n";

cout<<"\t"<<"\t"<<"\* Reshenie difurov 1 poryadka methodom Eulera \*"<<"\n";

cout<<"\t"<<"\t"<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

cout<<endl;

cout<<"Vvedite nachaloe znachenie intervala [a,b]=";

cin>>Xi;

cout<<"Vvedite konechoe znachenie intervala [a,b]=";

cin>>Xkon;

cout<<"Vvedite chislo shagov=";

cin>>n;

h=(Xkon- Xi)/n;

cout<<endl;

cout<<"Vvedite nachalnoe uslovie y=";

cin>>Yi;

cout<<"Vvedite koefitsient pri x=";

cin>>kx;

cout<<"Vvedite koefitsient pri y=";

cin>>ky;

cout<<"|Interval|Chislo shagov|Shag prirasheniya|Nacalnoe Y|Uravnenie vida:| "<<"\n";

cout<<"|--------|-------------|-----------------|----------|---------------|"<<"\n";

cout<<"|"<<"["<<Xi<<","<<Xkon<<"]" <<" |"<<n<<" |"<<h<<" |"<<Yi<<" |"<<"y'="<<kx<<"x"<<"+"<<ky<<"y"<<" |"<<"\n";

cout<<endl;

cout<<endl;

for (int i=1;i<=n;i++)

{

func(Xi,Yi,kx,ky,h);

cout<<"\n";

}

return 0;

}

void func(double& Xi, double& Yi, double kx, double ky, double h)

{

double f1,Yprom,a,Xprom;

f1=(kx\*Xi)+(ky\*Yi);

Yprom=Yi+f1\*(h/2);

Xprom=Xi+h/2;

a=kx\*Xprom-Yprom;

Yi=Yi+a\*h;

cout<<"\t"<<"\t"<<"Interval x="<<Xi<<"\t"<<" Resultat y="<<Yi;

Xi=Xi+h;

}