CАНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

**Лабораторная работа по выч.математике №4**

*«Решение обыкновенных дифференциальных уравнений.*

*Метод Адамса»*

Выполнил: Припадчев Артём

группа 2125

Проверил: Шипилов П.А.

2013 г.

**Задание:** составить подпрограмму для решения ОДУ первого порядка используя многошаговый метод Адамса. Разгонные точки вычислить методом Рунге-Кутта 4-го порядка. Вычисление правых частей реализовать отдельной подпрограммой. Найти решение заданного уравнения с точностью e, контролируя точность на каждом шаге вычислений, построить график решения.

**Описание методов**

**Метод Рунге-Кутты**

Воспользовавшись хорошо зарекомендовавшей себя формулой Симпсона, можно получить еще более точную формулу для решения задачи Коши для ОДУ первого порядка -  широко используемого в вычислительной практике метода Рунге-Кутты.

В формуле Симпсона для приближенного вычисления определенного интеграла используются значения подинтегрального выражения в  трех точках. В интеграле их всего две, поэтому введем дополнительную точку в середине отрезка [*xi+1 xi*]

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image086.gif

тогда можно переписать так:

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image088.gif

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image090.gif

Полученное выражение является неявным, так как в правой части содержатся  еще не определенные значения функции *yi+h/2* и *yi+1*. Чтобы воспользоваться этой формулой, надо использовать некоторое приближение для вычисления этих значений http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image092.gif http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image094.gif

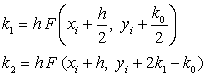
При использовании различных методов приближенного вычисления этих величин, получаются выражения для методов Рунге-Кутты различного порядка точности.

*Алгоритм Рунге-Кутты третьего порядка -***РК3***(погрешность порядка h3)*:

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image096.gif                                                                       (6.8)

где

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image098.gif

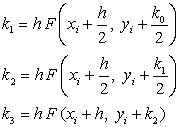


*Алгоритм Рунге-Кутты четвертого порядка-***РК4***(погрешность порядка h4)*:

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image102.gif                                                 (6.9)

где

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image104.gif



Алгоритмы третьего и четвертого порядков требуют на каждом шаге трех и четырех вычислений функции соответственно, но являются весьма точными.

**Метод Адамса**

Рассмотренный ранее метод Рунге-Кутты использует значение функции на одном предшествующем шаге, поэтому они относятся к так называемым *одношаговым методам*. Точность вычислений можно увеличить, если использовать при нахождении  решения в некотором узле *xi* информацию о значениях функции, полученных в *нескольких* (*k*) предыдущих узлах сетки интегрирования *(xi-1, xi-2 … xi-k*).

Если используются значения в *k* предыдущих узлах, то говорят о k-шаговом методе интегрирования уравнения. Одним из способов построения многошаговых методов заключается в следующем. По значениям функции, вычисленным в k предшествующих узлах, строится интерполяционный полином степени *(k-1)* - http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image122.gif, который используется при интегрировании дифференциального уравнения по выражению ([6.3](http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.html#6-3)). Интеграл при этом выражается через квадратурную формулу:

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/image124.gif,

где*λl* – квадратурные коэффициенты.

Очевидно, что при*k*=1 в качестве частного случая получается формула Эйлера. Значения квадратурных коэффициентов для *k* от 2 до 4 приведены в таблице.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *k* | *λl* | | | |
| 2 | 3/2 | -1/2 |  |  |
| 3 | 23/12 | -16/12 | 5/12 |  |
| 4 | 55/24 | -59/24 | 37/24 | -9/24 |

Полученное таким образом семейство формул называется *явной k-шаговой схемой Адамса*(*методы Адамса-Башфорта*).

Например, четырехшаговая явная формула Адамса может быть записана так:



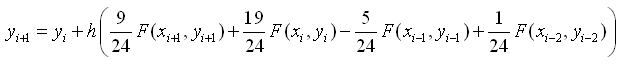
Если для построения интерполяционного полинома использовать k узлов, начиная с x*i*+1, то можно получить формулы интегрирования ОДУ, известные как неявные схемы Адамса (или методы *Адамса-Моултона*). Неявными эти формулы называются потому, что значение искомой функции в (i+1)-м узле -  y*i*+1 - оказывается одновременно и в левой и правой частях равенства.

http://www.physchem.chimfak.rsu.ru/Source/NumMethods/ODE.files/img17.gif

Квадратурные коэффициенты для неявных методов Адамса приведены в таблице ниже.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *k* | *λl* | | | |
| 2 | 1/2 | 1/2 |  |  |
| 3 | 5/12 | 8/12 | -1/12 |  |
| 4 | 9/24 | 19/24 | -5/24 | 1/24 |

Например, четырехшаговая неявная формула Адамса-Моултона имеет вид:



Видно, что это выражение является уравнением относительно  y*i*+1, так как   y*i*+1встречается и в левой и правой его части. Однако обычно это уравнение не решается, а значение в правой части заменяется на рассчитанное по какой-либо явной формуле - например, формуле Адамса Башфорта.

**Код программы**

public class DifferentialCalculate

{

Function function;

public double[] ResultXArr {get; private set; }

public double[] ResultYArr {get; private set; }

double xStart, x0, y0;

double xEnd;

public double h { get; private set; }

int steps;

double accuracy;

public DifferentialCalculate(Function function, double x0, double y0, double xStart, double xEnd, double h, double accuracy)

{

this.function = function;

this.xStart = xStart;

this.xEnd = xEnd;

this.h = h;

this.accuracy = accuracy;

this.x0 = x0;

this.y0 = y0;

}

private void RungeKutta()

{

for (int i = 1; i < 4; i++)

{

double k1 = function(ResultXArr[i - 1], ResultYArr[i - 1]);

double k2 = function(ResultXArr[i - 1] + h / 2, ResultYArr[i - 1] + k1 / 2);

double k3 = function(ResultXArr[i - 1] + h / 2, ResultYArr[i - 1] + k2 / 2);

double k4 = function(ResultXArr[i - 1] + h, ResultYArr[i - 1] + k3);

ResultYArr[i] = ResultYArr[i - 1] + h \* (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;

}

}

//Явная формула Адамса

private double ExplicitAdams(int i)

{

double yNext = ResultYArr[i - 1] + h / 24 \* (55 \* function(ResultXArr[i - 1], ResultYArr[i - 1]) - 59 \* function(ResultXArr[i - 2], ResultYArr[i - 2])

+ 37 \* function(ResultXArr[i - 3], ResultYArr[i - 3]) - 9 \* function(ResultXArr[i - 4], ResultYArr[i - 4]));

return yNext;

}

//Неявная формула Адамса

private double ImplicitAdams(int i)

{

double yNext = ResultYArr[i - 1] + h / 24 \* (9 \* function(ResultXArr[i], ResultYArr[i]) + 19 \* function(ResultXArr[i - 1], ResultYArr[i - 1])

- 5 \* function(ResultXArr[i - 2], ResultYArr[i - 2]) + 1 \* function(ResultXArr[i - 3], ResultYArr[i - 3]));

return yNext;

}

public void Calculate()

{

bool flag = false;

do

{

this.steps = (int)(Math.Round((xEnd - xStart) / h, 0) + 1);

ResultXArr = new double[steps];

ResultYArr = new double[steps];

this.ResultYArr[0] = y0;

for (int i = 0; i < steps; i++)

{

ResultXArr[i] = x0 + i \* h;

}

RungeKutta();

double currantAccuracy;

for (int i = 4; i < steps; i++)

{

ResultYArr[i] = ExplicitAdams(i);

currantAccuracy = Math.Abs(ResultYArr[i] - ImplicitAdams(i));

if (currantAccuracy > accuracy)

{

flag = false;

break;

}

else

flag = true;

}

h = h / 2;

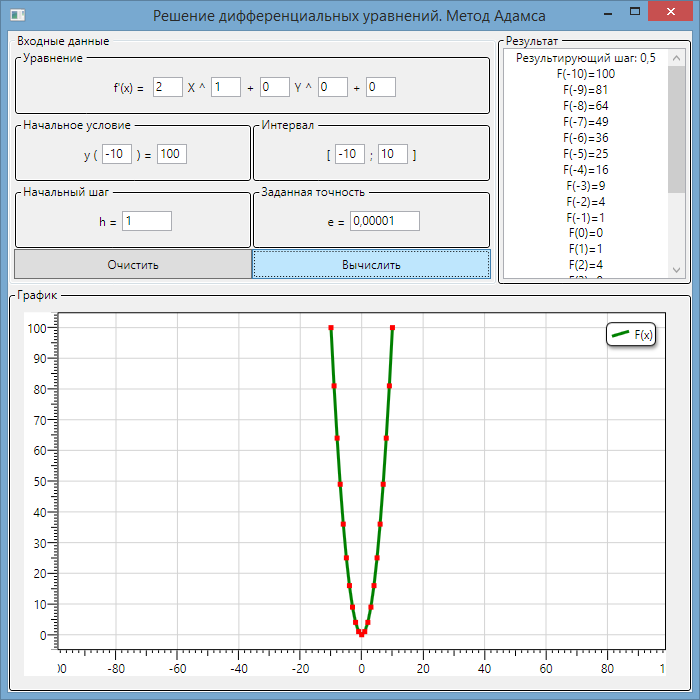
}

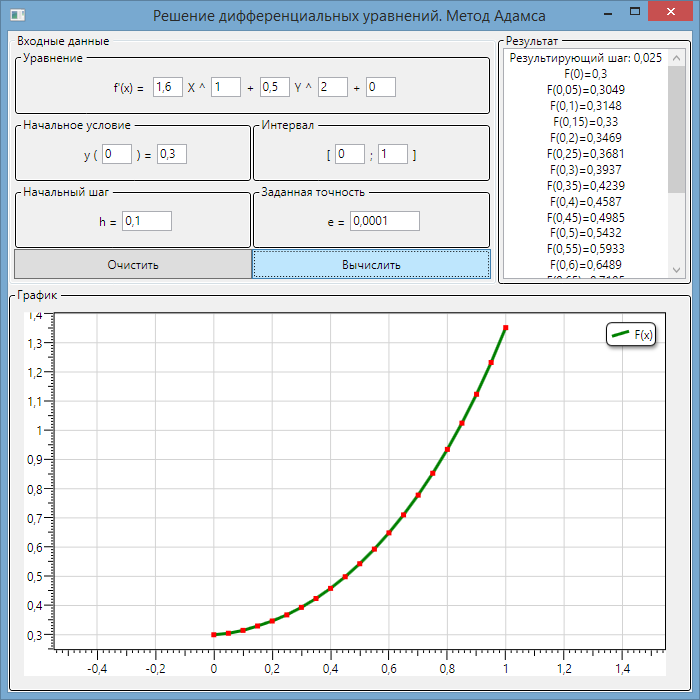
while (!flag);

}

}

**Примеры работы**





**Вывод:** в процессе выполнения лабораторной работы были рассмотрены методы Рунге-Кутты и Адамса для решения обыкновенных дифференциальных уравнений.

Достоинством многошаговых методов Адамса при решении ОДУ заключается в том, что в каждом узле рассчитывается только одно значение правой части ОДУ - функции *F(x,y*). К недостаткам можно отнести невозможность старта многошагового метода из единственной начальной точки, так как для вычислений по *k*-шаговой формуле необходимо знание значения функции в *k* узлах. Поэтому приходится *(k-1)* решение в первых узлах *x1, x2, …, xk-1* получать с помощью какого-либо одношагового метода, например метода Рунге-Кутты 4–го порядка.

Другой проблемой является невозможность изменения шага в процессе решения, что легко реализуется в одношаговых методах.