Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Дисциплина: Вычислительная математика

Лабораторная работа №4

«Решение обыкновенных дифференциальных уравнений (задачи Коши) методом Милна»

Выполнил Борзых А.А, гр. P3217

2015г.

Описание метода

Метод Рунге-Кутта 4-го порядка

Рассмотрим задачу Коши



Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:



Вычисление нового значения проходит в четыре стадии:



где  — величина шага сетки по .



Этот метод имеет четвёртый порядок точности, то есть суммарная ошибка на конечном интервале интегрирования имеет порядок  (ошибка на каждом шаге порядка )



Метод Милна

Метод Милна относится к многошаговым методам и представляет один из методов прогноза и коррекции. Решение в следующей точке находится в два этапа. На первом этапе осуществляется по специальной формуле прогноз значения функции, а затем на втором этапе - коррекция полученного значения. Если полученное значение у после коррекции существенно отличается от спрогнозированного, то проводят еще один этап коррекции. Если опять имеет место существенное отличие от предыдущего значения (т.е. от предыдущей коррекции), то проводят еще одну коррекцию и т.д. Однако очень часто ограничиваются одним этапом коррекции.

Метод Милна имеет следующие вычислительные формулы:

а) этап прогноза:



где для компактности записи использовано следующее обозначение

fi = f(xi, yi);

б) этап коррекции:



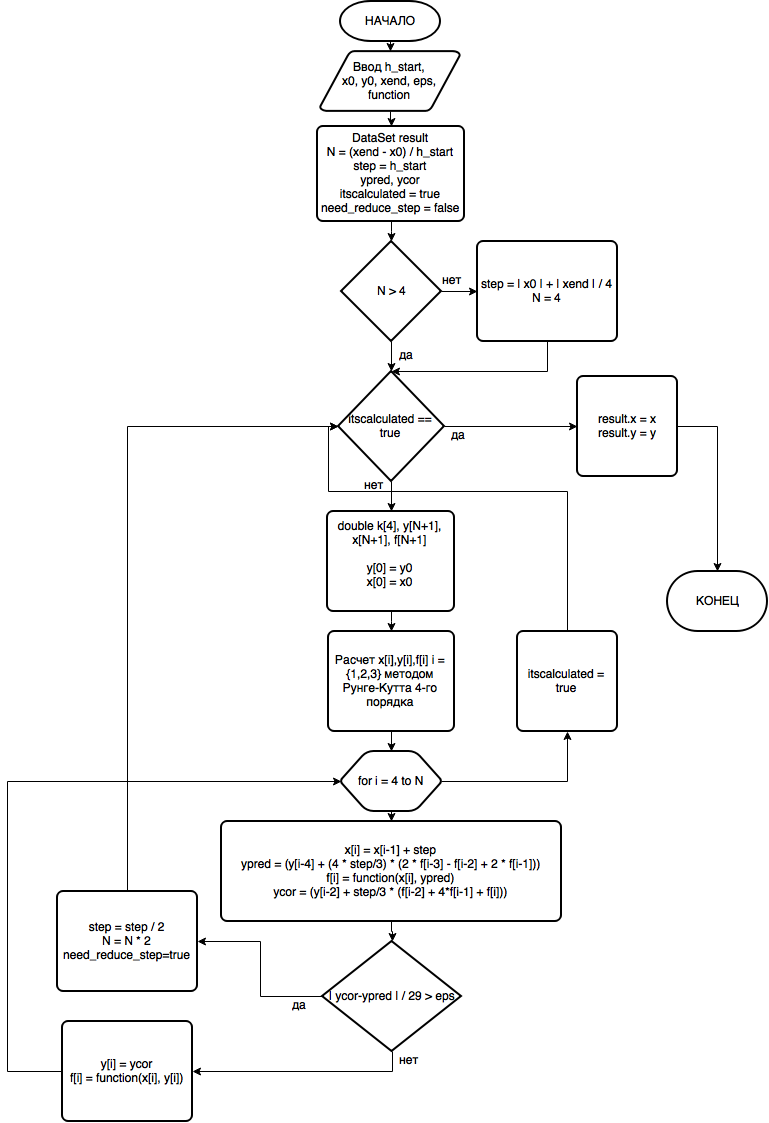
Абсолютная погрешность определяется по формуле



Для своей работы метод Милна требует начальный отрезок значений.

В рамках данной лабораторной работы первые 3 значения Y были получены при помощи метода Рунге-Кутта 4-го порядка, последующие при помощи метода Милна.

Блок схема алгоритма



Листинг метода

DataSet Milnes::Milnes\_Method(double x0, double y0, double xn, double eps,

std::function<double(double,double)> func, double start\_step)

{

DataSet result; // stores arrays of x,y

double step = start\_step;

double n\_points = (xn - x0) / start\_step; // number of points to be returned

n\_points = std::floorf(n\_points);

if(n\_points < 4)// n\_points can't be less than 4

{

step = (std::abs(x0) + std::abs(xn)) / 4;

n\_points = 4;

}

double y\_pred, y\_cor; // y\_pred - y predictor, y\_cor - y corrector

bool its\_calculated = false;

bool need\_reduce\_step = false;

while(!its\_calculated)

{

need\_reduce\_step = false;

its\_calculated = true;

QVector<double> k(4);

QVector<double> y(n\_points + 1);

QVector<double> x(n\_points + 1);

QVector<double> f(n\_points + 1);

y[0] = y0;

x[0] = x0;

// calculate first 4 elements using Runge-Kutta formulas

for(auto i = 0; i < 3; i++)

{

k[0] = func(x[i], y[i]);

k[1] = func(x[i] + step/2, y[i] + k[0] / 2);

k[2] = func(x[i] + step/2, y[i] + k[1] / 2);

k[3] = func(x[i] + step, y[i] + k[2]);

y[i + 1] = (y[i] + step \* (k[0] + 2 \* k[1] + 2 \* k[2] + k[3])/6.0);

x[i + 1] = x[i] + step;

f[i] = (func(x[i], y[i]));

}

f[3] = func(x[3], y[3]);

// calculate others using Milne's formulas

for(auto i = 4; i <= n\_points; i++)

{

x[i] = x[i - 1] + step;

y\_pred = (y[i-4] + (4 \* step/3) \* (2 \* f[i - 3] - f[i - 2] + 2 \* f[i - 1]));

f[i] = (func(x[i], y\_pred));

y\_cor = (y[i - 2] + step / 3 \* (f[i - 2] + 4 \* f[i - 1] + f[i]));

if(std::abs((y\_cor - y\_pred)/29) > eps) // check accuracy

{

step = step / 2; // reduce step

n\_points = n\_points \* 2; // and increase n of points

need\_reduce\_step = true;

break;

}

else

{

y[i] = y\_cor;

f[i] = func(x[i], y[i]);

}

}

if(need\_reduce\_step) its\_calculated = false;

result.x = x;

result.y = y;

result.lenght = x.size();

}

return result;

}

Пример работы программы

Исходная функция y = x^2

y’(x) = 3 \* x^2

x0 = -3, xn = 3, eps = 0.01, start\_step = 0.5, y(x0) = -27

Полученные значения:

|  |  |  |
| --- | --- | --- |
| x | Y (calculated | y (real) |
| -3 | -27 | -27 |
| -2,5 | -15,625 | -15,625 |
| -2 | -8 | -8 |
| -1,5 | -3,375 | -3,375 |
| -1 | -1 | -1 |
| 0 | 0 | 0 |
| 0,5 | 0,125 | 0,125 |
| 1 | 1 | 1 |
| 1,5 | 3,375 | 3,375 |
| 2 | 8 | 8 |
| 2,5 | 15,625 | 15,625 |
| 3 | 27 | 27 |

Вывод

В процессе выполнения лабораторной работы был рассмотрен Метод Рунге-Кутты и метод Милна для решения обыкновенных дифференциальных уравнений. Метод Милна обладает достаточной точностью при решении ОДУ.

Метод требует несколько меньшего количества вычислений (например, достаточно только два раза вычислить f(x, y), остальные запомнены с предыдущих этапов), но требует дополнительного "расхода" памяти.

Однако для своей работы метод требует некоторого количества уже найденных точек (4 и более) для своей работы.