МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ

УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА

Институт радиоэлектроники и информационных технологий

Кафедра информатики и систем управления

Приближенное решение обыкновенных дифференциальных уравнений

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к лабораторной работе №6

по дисциплине

Вычислительная Математика

РУКОВОДИТЕЛЬ:

Суркова Анна Сергеевна

____

(подпись)

СТУДЕНТ:

Цветков Николай Максимович

____

(подпись)

19-ИВТ-3

Работа защищена «___» ____________

С оценкой ________________________

Нижний Новгород 2021

Оглавление

Цель

Закрепление знаний и умений по численному решению обыкновенных дифференциальных уравнений методом Эйлера, методом Эйлера с пересчетом, методом Рунге-Кутты и методом Адамса.

Постановка задачи

Задание 1

Используя метод Эйлера и метод Эйлера с пересчетом, составить таблицу приближенных значений интеграла дифференциального уравнения $y' = f(x, y)$, удовлетворяющих начальным условиям $y(x_0) = y_0$ на отрезке $[a, b]$; шаг $h = 0.1$. Все вычисление вести с четырехзначными знаками. Проверить полученные значения, использую метод Рунге-Кутты 4 порядка.

$$4. y' = x + cos\frac{y}{\sqrt{7}} \qquad y_0(0.5) = 0.6, \;\; x \in [0.5; 1.5]$$

Задание 2

Используя метод Адамса с третьими разностями составить таблицу приближенных значений интеграла дифференциального уравнения $y' = f(x, y)$, удовлетворяющих начальным условиям $y(x_0) = y_0$ на отрезке $[0, 1]$; шаг $h = 0.1$. Все вычисление вести с четырехзначными знаками. Начальный отрезок определить методом Рунге-Кутты. Проверить полученные значения, используя метод Эйлера с пересчетом.

$$4. y' = (1 - y^2)\cos(x) + 0.6(y) \qquad y(0) = 0$$

Теоретические сведения

Метод Эйлера и метод Эйлера с пересчетом

Обозначим границы отрезка а и Ь через x0 и xN соответственно и введем на отрезке [а, ?>] сетку (в общем случае неравномерную) значений аргумента x такую, чтобы выполнялось соотношение x0 < x1 < ... < xN. Выполним разложение решения и (л:) в окрестности узла сетки хп по формуле Тейлора. Обозначив шаг сетки через hn = хп + 1- хп и и(хп) через ип, получаем

Если функция f имеет непрерывную р-ю производную, то в соотношении (3.15) можно оставлять члены вплоть до 0(hP ' *). Эти производные можно найти, дифференцируя правую часть уравнения (3.14) требуемое число раз. Например, для первой и второй производных имеем

$$u'(x) = f(x, u),$$
$$u''(x) = \frac{d}{dx}u'(x) = \frac{d}{dx}f(x, u) = f'_u u' + f'_x = ff'_u + f'_x = f\frac{\partial f}{\partial u} + \frac{\partial f}{\partial x}.$$

$$u_{n+1} = u_n + h_n u'_n + \frac{1}{2!}h_n^2 u''_n + \frac{1}{3!}h_n^3 u'''_n + \ldots . \qquad (3.15)$$

налогично можно получить производные более высоких порядков.

Однако использование формулы (3.15) с большим числом членов имеет ряд недостатков: во-первых, с ростом порядка производной выражение для нее может оказаться очень сложным; кроме того, если функция f известна лишь приближенно или задана таблично, ее производные находятся с большой ошибкой. В связи с этим в разложении (3.15) оставляют только два члена. При такой замене вместо точного решения и{хп т j) получается его приближенное значение и(хп + t), которое находится по формуле

$$u_{n+1} = u_n + h_n f(x_n, u_n). \qquad (3.16)$$

Так как значение п(дс0) = ь то, последовательно пользуясь женные решения uv и2, ..., uN,

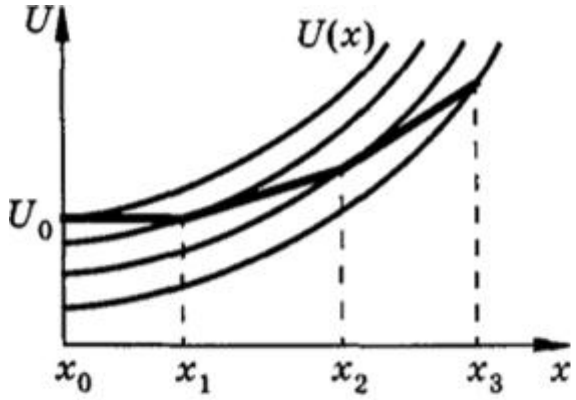


Рис. 3.1

0 известно из начального условия, формулой (3.16), находим прибли-
Формула (3.16) записана для случая неравномерной сетки. Полагая шаг сетки hn = h постоянным, получим

$$u_{n+1} = u_n + hf(x_n, u_n). \quad (3.16a)$$

Формула (3.16а) является ОСНОВНОЙ ФОРМУЛОЙ МЕТОДА ЭЙЛЕРА или МЕТОДА ЛОМАНЫХ.

Существует несколько модификаций метода Эйлера. Остановимся на одной из них — методе Эйлера с пересчетом, который называют также МЕТОДОМ ЭЙЛЕРА — КОШИ. В этом методе значение ип _ j находится по формуле

$$u_{n+1} = u_n + \frac{h}{2}[f(x_n, u_n) + f(x_{n+1}, u_{n+1})], \qquad (3.17)$$

т. е. вместо тангенса угла наклона касательной к интегральной кривой в точке (хп, ип), который имеет место в формуле (3.16а), выражающей метод Эйлера, используется полусумма значений тангенсов углов наклона касательных в известной (хп, ип) и искомой (хп + j, ип + t) точках. Поскольку, однако, значение ип + г неизвестно, то (3.17) есть в общем случае нелинейное уравнение относительно ип + 1, которое можно решить различными методами, изложенными в главе 1.

В рассматриваемом случае логично использовать метод простой итерации, представленный формулой (1.10), поскольку нелинейное уравнение уже разрешено относительно ип + v Тогда, если номер итерации обозначить верхним индексом, итерационный процесс запишется в виде

$$u_{n+1}^{(k+1)} = u_n + \frac{h}{2}[f(x_n, u_n) + f(x_n + h, u_{n+1}^{(k)})]. \qquad (3.18)$$

В качестве значения и(л0)+1 можно принять либо ип, либо и^+1 = ~ ип + hf(xn, ип), т. е. использовать значение, вычисленное по формуле Эйлера (3.16а). В этом случае в первой итерации имеем

$$u_{n+1}^{(1)} = u_n + \frac{h}{2}\{f(x_n, u_n) + f\,[(x_n + h, u_n + hf(x_n, u_n)]\}. \qquad (3.19)$$

Формула (3.19) и есть ОСНОВНАЯ ФОРМУЛА МЕТОДА ЭЙЛЕРА С ПЕРЕСЧЕТОМ. Подобные схемы часто называют схемами типа ПРОГНОЗ-КОРРЕКТОР (ПРЕДИКТОР-КОРРЕКТОР). Сначала по формуле (3.16) определяется прогнозируемое приближение решения, а затем по формуле (3.19) это решение уточняется.

Метод Эйлера с пересчетом обладает третьим порядком точности на шаге. Действительно, из соотношения (3.15) следует

$$u_{n+1} = u_n + hf(x_n, u_n) + \frac{h^2}{2}\left[\frac{\partial f}{\partial x} + f(x_n, u_n)\frac{\partial f}{\partial u}\right] + O(h^3). \qquad (3.20)$$

Разлагая в ряд второй член в квадратных скобках (3.19), имеем

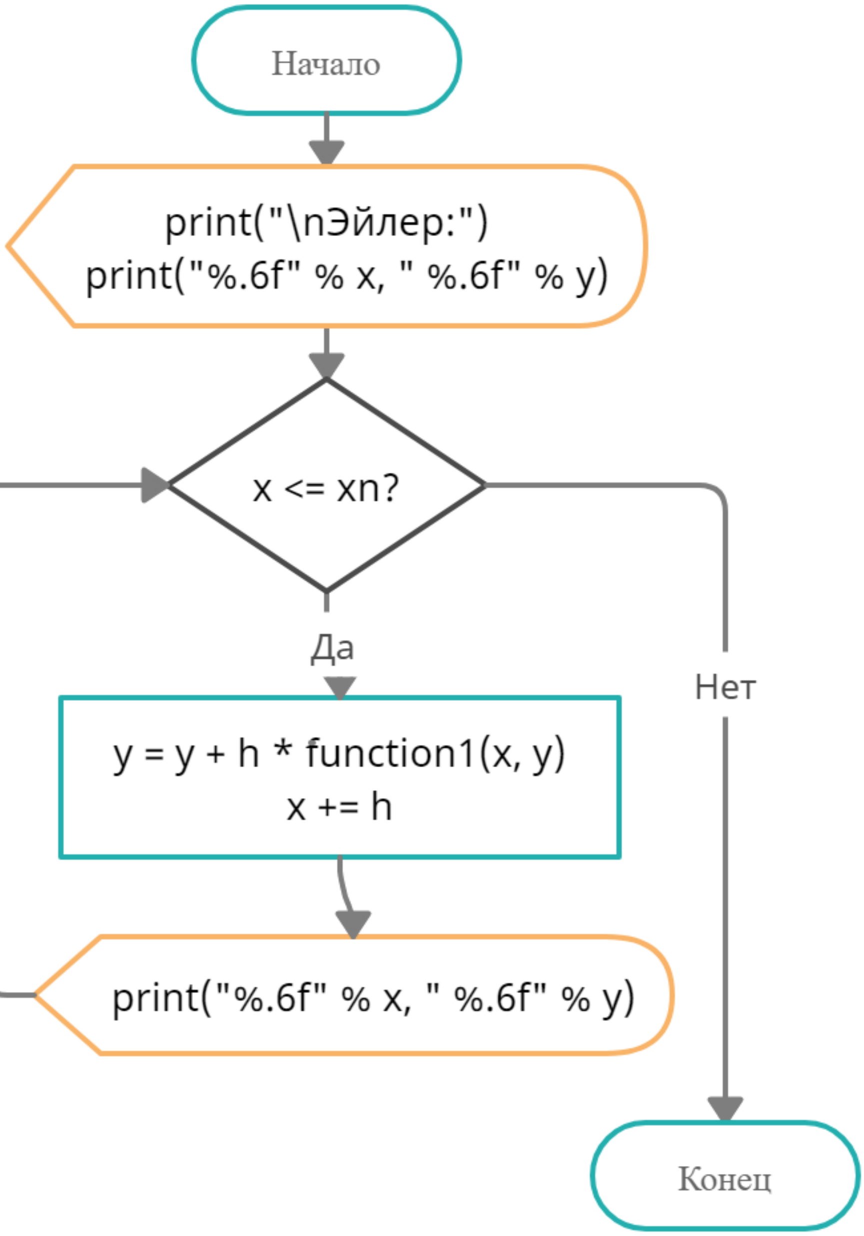
$$f[x_n + h, u_n + hf(x_n, u_n)] = f(x_n, u_n) + \frac{\partial f}{\partial x}h + \frac{\partial f}{\partial u}hf(x_n, u_n). \qquad (3.21)$$

Подставляя правую часть соотношения (3.21) в последний член выражения (3.19), получаем

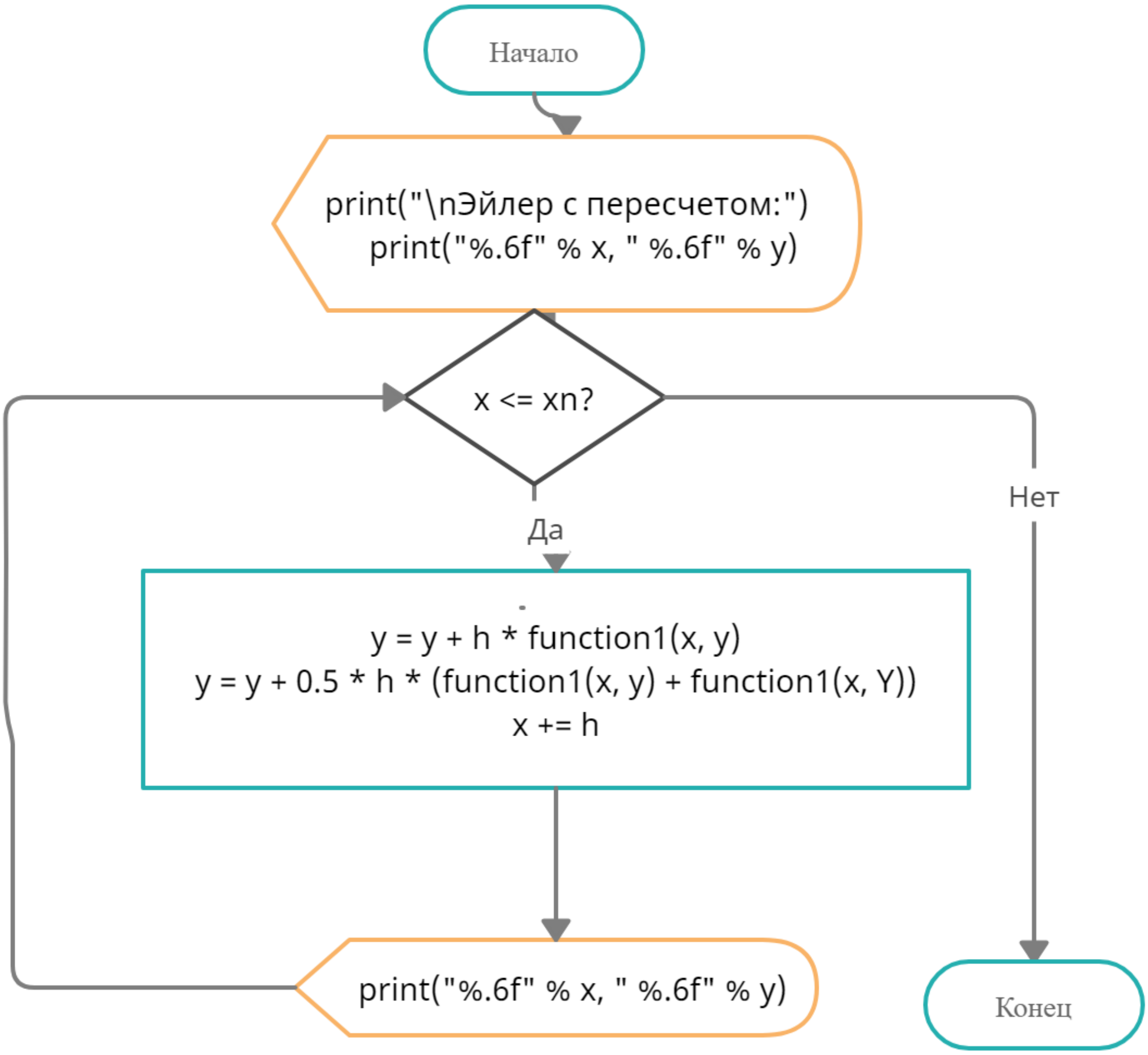
$$u_{n+1} = u_n + hf(x_n, u_n) + \frac{h^2}{2}\left[\frac{\partial f}{\partial x} + f(x_n, u_n)\frac{\partial f}{\partial u}\right]. \qquad (3.22)$$

Сравнивая соотношения (3.20) и (3.22), видим, что метод Эйлера с пересчетом обладает третьим порядком точности на шаге и, соответственно, вторым на интервале. Метод Эйлера с пересчетом дает двустороннее приближение к решению.

Эйлер


Начало
print("\nЭйлер:")
print("%.6f" % x, " %.6f" % y)
x <= xn?
Да
Нет
y = y + h * function1(x, y)
x += h
print("%.6f" % x, " %.6f" % y)
Конец

Эйлер с пересчетом



Метод Рунге-Кутты

Метод Рунге-Кутты используют для расчета стандартных моделей достаточно часто, так как при небольшом объеме вычислений он обладает точностью метода O4(h).

Для построения разностной схемы интегрирования воспользуемся разложением функции

$$\frac{dy}{dx} = f(x, y(x)),\ 0 < x \le T,\ y(0) = y_0 \qquad (1)$$

в ряд Тейлора:

$$y(x_{k+1}) = y(x_k) + y'(x_k)h + y''(x_k)\frac{h^2}{2} + \ldots$$

Заменим вторую производную в этом разложении выражением

$$y''(x_k) = (y'(x_k))' = f'(x_k, y(x_k)) \approx \frac{f(\tilde{x}, \tilde{y}) - f(x_k, y(x_k))}{\Delta x}$$

где

$$\tilde{x} = x_k + \Delta x,\ \tilde{y} = y(x_k + \Delta x)$$

Причем Δx подбирается из условия достижения наибольшей точности записанного выражения. Для дальнейших выкладок произведем замену величины «y с тильдой» разложением в ряд Тейлора:

$$\tilde{y} = y(x_k + \Delta x) = y(x_k) + y'(x_k)\Delta x + \ldots$$

Для исходного уравнения (1) построим вычислительную схему:

$$y_{k+1} = y_k + f(x_k, y_k)\cdot h + \frac{h^2}{2\Delta x}\cdot(f(x_k + \Delta x,\ y_k + y_k'\Delta x) - f(x_k, y_k))$$

которую преобразуем к виду:

$$y_{k+1} = y_k + h\cdot\left[\left(1 - \frac{h}{2\Delta x}\right)\cdot f(x_k, y_k) + \frac{h}{2\Delta x}f(x_k + \Delta x,\ y_k + y_k'\Delta x)\right] =$$
$$= y_k + h\cdot\left[\left(1 - \frac{h}{2\Delta x}\right)\cdot f(x_k, y_k) + \frac{h}{2\Delta x}f(x_k + \frac{\Delta x}{h}h,\ y_k + f(x_k, y_k)\frac{\Delta x}{h}h)\right]$$

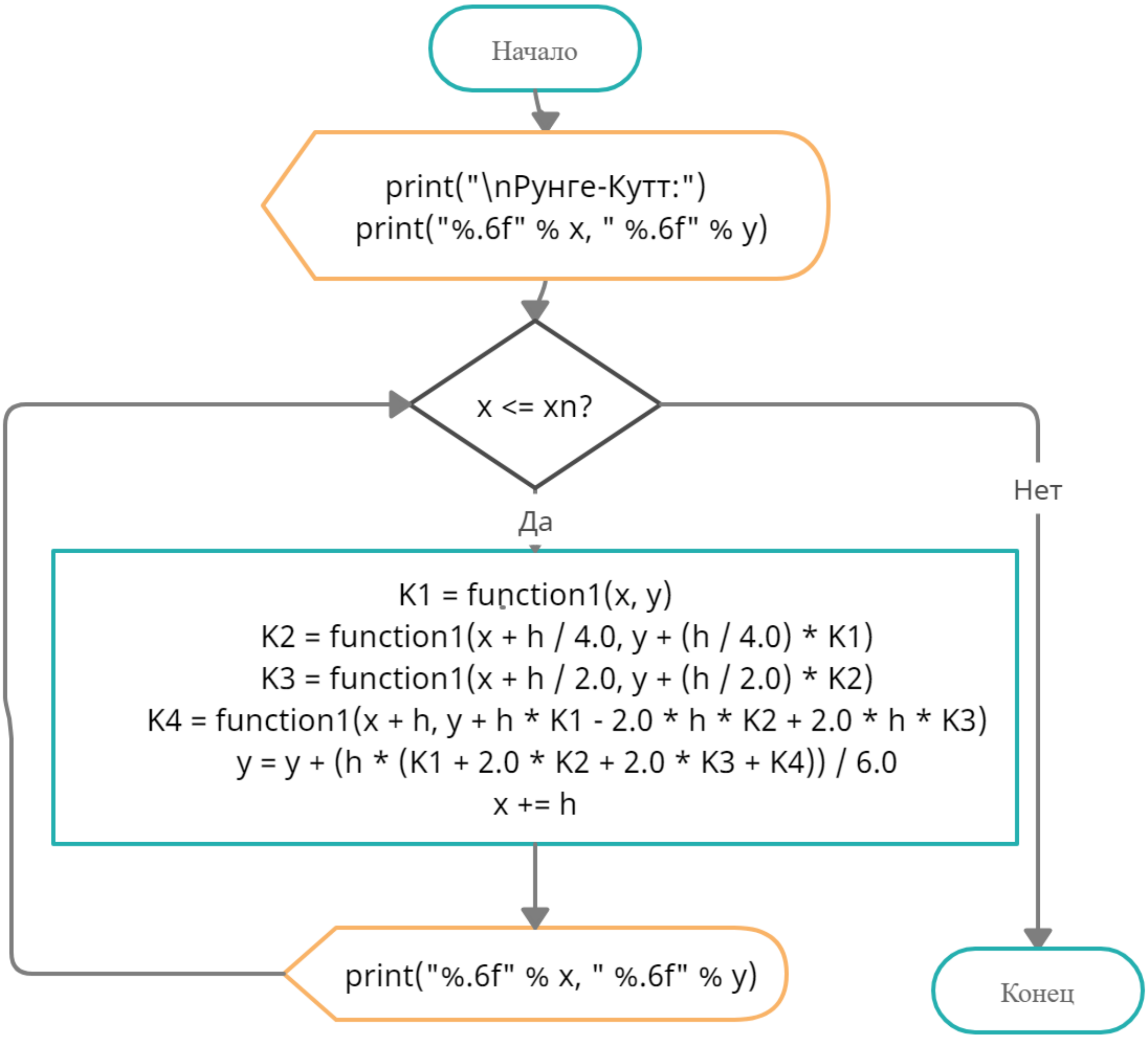
Введем следующие обозначения:

$$\alpha = \frac{h}{2\Delta x},\ \beta = 1 - \frac{h}{2\Delta x},\ \gamma = \frac{\Delta x}{h},\ \delta = f(x_k, y_k)\frac{\Delta x}{h}$$

Эти обозначения позволяют записать предыдущее выражение в форме:

$$y_{k+1} = y_k + h\cdot\left[\beta\cdot f(x_k, y_k) + \alpha\cdot f(x_k + \gamma\cdot h,\ y_k + \delta\cdot h)\right]$$

Рунге-Кутт


Начало
print("\nРунге-Кутт:")
print("%.6f" % x, " %.6f" % y)
x <= xn?
Да
Нет
K1 = function1(x, y)
K2 = function1(x + h / 4.0, y + (h / 4.0) * K1)
K3 = function1(x + h / 2.0, y + (h / 2.0) * K2)
K4 = function1(x + h, y + h * K1 - 2.0 * h * K2 + 2.0 * h * K3)
y = y + (h * (K1 + 2.0 * K2 + 2.0 * K3 + K4)) / 6.0
x += h
print("%.6f" % x, " %.6f" % y)
Конец

Метод Адамса

7. Метод Адамса. Будем рассматривать правую часть уравнения $f(x, u)$ не на всей плоскости ее аргументов x, u, а только на определенной интегральной кривой $u(x)$, соответствующей искомому решению. Тогда она будет функцией только одного аргумента x; обозначим ее через

$$F(x) \equiv f(x, u(x)).$$

Пусть нам уже известно приближенное решение в нескольких точках сетки: $y_n, y_{n-1}, \ldots, y_{n-m}$. Тогда в этих точках известны также $F(x_k) = f(x_k, y_k)$. В окрестности этих узлов можно приближенно заменить $F(x)$ интерполяционным многочленом; запишем его для неравномерной сетки в форме Ньютона (2.8):

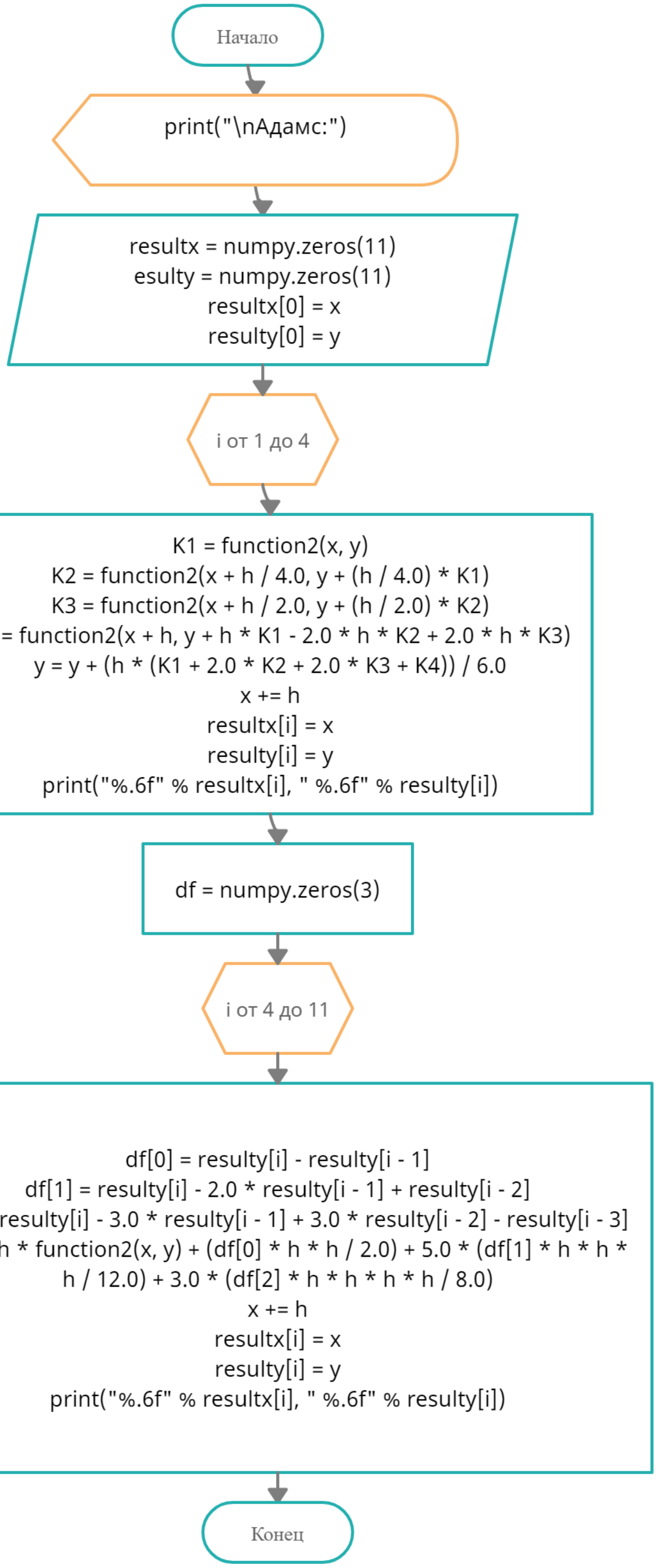
$$F(x) = F(x_n) + (x - x_n) F(x_n, x_{n-1}) + \\ + (x - x_n)(x - x_{n-1}) F(x_n, x_{n-1}, x_{n-2}) + \\ + (x - x_n)(x - x_{n-1})(x - x_{n-2}) F(x_n, x_{n-1}, x_{n-2}, x_{n-3}) + \ldots \quad (28)$$

к формулам (31). Все это делает метод Адамса неудобным для расчетов на ЭВМ.

Внешне этот метод привлекателен тем, что за один шаг приходится только один раз вычислять $f(x, u)$, которая может быть очень сложной. А в четырехчленной схеме Рунге — Кутта того же порядка точности $f(x, u)$ вычисляется за шаг четыре раза. Однако коэффициент в остаточном члене (27) схемы Рунге — Кутта (24) меньше в 960 раз, чем в схеме (31)! Значит, при одинаковой точности схема Рунге — Кутта (24) позволяет брать шаг в $\sqrt[4]{960} = 5{,}7$ раза крупнее, т. е. фактически вычислять $f(x, u)$ даже меньшее число раз, чем в методе Адамса.

Поэтому сейчас метод Адамса и аналогичные методы (например, Милна) употребляются реже метода Рунге — Кутта.

Адамс


Начало
print("\nАдамс:")
resultx = numpy.zeros(11)
esulty = numpy.zeros(11)
resultx[0] = x
resulty[0] = y
i от 1 до 4
K1 = function2(x, y)
K2 = function2(x + h / 4.0, y + (h / 4.0) * K1)
K3 = function2(x + h / 2.0, y + (h / 2.0) * K2)
K4 = function2(x + h, y + h * K1 - 2.0 * h * K2 + 2.0 * h * K3)
y = y + (h * (K1 + 2.0 * K2 + 2.0 * K3 + K4)) / 6.0
x += h
resultx[i] = x
resulty[i] = y
print("%.6f" % resultx[i], " %.6f" % resulty[i])
df = numpy.zeros(3)
i от 4 до 11
df[0] = resulty[i] - resulty[i - 1]
df[1] = resulty[i] - 2.0 * resulty[i - 1] + resulty[i - 2]
df[2] = resulty[i] - 3.0 * resulty[i - 1] + 3.0 * resulty[i - 2] - resulty[i - 3]
y = y + h * function2(x, y) + (df[0] * h * h / 2.0) + 5.0 * (df[1] * h * h *
h / 12.0) + 3.0 * (df[2] * h * h * h * h / 8.0)
x += h
resultx[i] = x
resulty[i] = y
print("%.6f" % resultx[i], " %.6f" % resulty[i])
Конец

Расчетные данные

Задание 1

Метод Эйлера

X	Y
0.500000	0.600000
0.600000	0.747440
0.700000	0.903476
0.800000	1.067702
0.900000	1.239669
1.000000	1.418891
1.100000	1.604852
1.200000	1.797012
1.300000	1.994820
1.400000	2.197717
1.500000	2.405156

Метод Эйлера с пересчетом

X	Y
0.500000	0.600000
0.600000	0.746738
0.700000	0.901877
0.800000	1.064995
0.900000	1.235633
1.000000	1.413297
1.100000	1.597471
1.200000	1.787619
1.300000	1.983202
1.400000	2.183677
1.500000	2.388520

Метод Рунге-Кутта

X	Y
0.500000	0.600000
0.600000	0.751032
0.700000	0.910451
0.800000	1.077820
0.900000	1.252664
1.000000	1.434474
1.100000	1.622717
1.200000	1.816845
1.300000	2.016302

1.400000	2.220536
1.500000	2.429010

Задание 2

Метод Адамса

X	Y
0.100000	0.102123
0.200000	0.207621
0.300000	0.313531
0.400000	0.416727
0.500000	0.515522
0.600000	0.608032
0.700000	0.693177
0.800000	0.770682
0.900000	0.840974
1.000000	0.905013

Метод Эйлера с пересчетом

X	Y
0.100000	0.102500
0.200000	0.208632
0.300000	0.315352
0.400000	0.419767
0.500000	0.519484
0.600000	0.612846
0.700000	0.699013
0.800000	0.777913
0.900000	0.850111
1.000000	0.916652

Листинг разработанной программы

Main.py

```
from solutionMethods import *

print("\nЗадание №1")
eulerMethod(x = 0.5, xn = 1.5, y = 0.6, h = 0.1)
eulerMethodWithRecalculation1(x = 0.5, xn = 1.5, y = 0.6, h = 0.1)
rungeKuttMethod(x = 0.5, xn = 1.5, y = 0.6, h = 0.1)
print("\nЗадание №2")
adamsMethod(x = 0.0, xn = 1.0, y = 0.0, h = 0.1)
eulerMethodWithRecalculation2(x = 0.0, xn = 1.0, y = 0.0, h = 0.1)
```

solutionMethods.py

```
import math
import numpy

def function1(x, y) -> float: # Функция для первого задания
    return x + math.cos(y / math.sqrt(7))
def function2(x, y) -> float: # Функция для второго задания
    return (1 - y**2) * math.cos(x) + 0.6 * y
def eulerMethod(x, xn, y, h):
    print("\nЭйлер:")
    print("%.6f" % x, " %.6f" % y) # Показываем первые значения
    while x <= xn: # Вычисляем все остальные точки по формуле
        y = y + h * function1(x, y)
        x += h
        print("%.6f" % x, " %.6f" % y)
def eulerMethodWithRecalculation1(x, xn, y, h):
    print("\nЭйлер с пересчётом:")
    print("%.6f" % x, " %.6f" % y) # Показываем первые значения
    while x <= xn: # Вычисляем все остальные точки по формуле
        Y = y + h * function1(x, y)
        y = y + 0.5 * h * (function1(x, y) + function1(x, Y))
        x += h
        print("%.6f" % x, " %.6f" % y)
def rungeKuttMethod(x, xn, y, h):
    print("\nРунге-Кутт:")
    print("%.6f" % x, " %.6f" % y) # Показываем первые значения
    while x <= xn: # Вычисляем все остальные точки по формуле
        K1 = function1(x, y)
        K2 = function1(x + h / 4.0, y + (h / 4.0) * K1)
        K3 = function1(x + h / 2.0, y + (h / 2.0) * K2)
        K4 = function1(x + h, y + h * K1 - 2.0 * h * K2 + 2.0 * h * K3)
        y = y + (h * (K1 + 2.0 * K2 + 2.0 * K3 + K4)) / 6.0
        x += h
        print("%.6f" % x, " %.6f" % y)
def adamsMethod(x, xn, y, h):
    print("\nАдамс:")
    resultx = numpy.zeros(11) # Здесь мы храним результаты
```

```
    resulty = numpy.zeros(11)
    resultx[0] = x # Добавляем в массив начальные значения
    resulty[0] = y
    for i in range(1, 4): # Считаем начальный отрезок методом Рунге-Кутта
        K1 = function2(x, y)
        K2 = function2(x + h / 4.0, y + (h / 4.0) * K1)
        K3 = function2(x + h / 2.0, y + (h / 2.0) * K2)
        K4 = function2(x + h, y + h * K1 - 2.0 * h * K2 + 2.0 * h * K3)
        y = y + (h * (K1 + 2.0 * K2 + 2.0 * K3 + K4)) / 6.0
        x += h
        resultx[i] = x
        resulty[i] = y
        print("%.6f" % resultx[i], " %.6f" % resulty[i])
    df = numpy.zeros(3) # Считаем все остальные значения при помощи метода Адамса
    for i in range(4, 11):
        df[0] = resulty[i] - resulty[i - 1]
        df[1] = resulty[i] - 2.0 * resulty[i - 1] + resulty[i - 2]
        df[2] = resulty[i] - 3.0 * resulty[i - 1] + 3.0 * resulty[i - 2] - result
y[i - 3]
        y = y + h * function2(x, y) + (df[0] * h * h / 2.0) + 5.0 * (df[1] * h *
h * h / 12.0) + 3.0 * (df[2] * h * h * h * h / 8.0)
        x += h
        resultx[i] = x
        resulty[i] = y
        print("%.6f" % resultx[i], " %.6f" % resulty[i])


def eulerMethodWithRecalculation2(x, xn, y, h):
    print("\nЭйлер с пересчётом:")
    while x < xn-h: # Вычисляем точки по формуле
        Y = y + h * function2(x, y)
        y = y + 0.5 * h * (function2(x, y) + function2(x, Y))
        x += h
        print("%.6f" % x, " %.6f" % y)
```

Результаты работы программы

```
Задание №1

Эйлер:
0.500000  0.600000
0.600000  0.747440
0.700000  0.903476
0.800000  1.067702
0.900000  1.239669
1.000000  1.418891
1.100000  1.604852
1.200000  1.797012
1.300000  1.994820
1.400000  2.197717
1.500000  2.405156

Эйлер с пересчётом:
0.500000  0.600000
0.600000  0.746738
0.700000  0.901877
0.800000  1.064995
0.900000  1.235633
1.000000  1.413297
1.100000  1.597471
1.200000  1.787619
1.300000  1.983202
1.400000  2.183677
1.500000  2.388520

Рунге-Кутт:
0.500000  0.600000
0.600000  0.751032
0.700000  0.910451
0.800000  1.077820
0.900000  1.252664
1.000000  1.434474
1.100000  1.622717
1.200000  1.816845
1.300000  2.016302
1.400000  2.220536
1.500000  2.429010
```

```
Задание №2

Адамс:
0.100000  0.102123
0.200000  0.207621
0.300000  0.313531
0.400000  0.416727
0.500000  0.515522
0.600000  0.608032
0.700000  0.693177
0.800000  0.770682
0.900000  0.840974
1.000000  0.905013

Эйлер с пересчётом:
0.100000  0.102500
0.200000  0.208632
0.300000  0.315352
0.400000  0.419767
0.500000  0.519484
0.600000  0.612846
0.700000  0.699013
0.800000  0.777913
0.900000  0.850111
1.000000  0.916652
```

Вывод

В ходе данной работы были закреплены знания и умения по вычислению приближенных значений интеграла дифференциального уравнения при помощи методом Эйлера, Эйлера с пересчетом, Рунге-Кутты и Адамса.