# Лабораторная работа №2

# Решение обыкновенных дифференциальных уравнений (часть 2)

## 1. Методы решения ОДУ

Для решения ОДУ в данной лабораторной работе будут использованы: методы Адамса-Башфорта и Адамса-Мултона 2-го порядка, метод Гира 2-го порядка.

При построении численных алгоритмов будем считать, что решение этой дифференциальной задачи существует, оно единственно и обладает необходимыми свойствами гладкости.

Идею численных методов решения задачи возьмем из предыдущей лабораторной работы (Лабораторная работ №1).

### Метод Адамса

**Метод Адамса** — конечноразностый многошаговый метод численного интегрирования дифференциальных уравнений первого порядка. Для вычисления очередного значения искомого решения использует не одно, а несколько значений, которые уже вычислены в предыдущих точках.

Существует несколько расчетных формул метода Адамса для решения системы дифференциальных уравнений певрого порядка:

1. Экстраполяционные — метод Адамса-Башфорта
2. Интерполяционные или неявные — метод Адамса-Мултона

### Метод Адамса-Башфорта

Явные методы Адамса-Башфорта:

- 1-ый порядок:

$$y_{n+1} = y_n + \tau F(t_n, y_n)$$

- 2-ой порядок:

$$y_{n+2} = y_{n+1} + \tau \left( \frac{3}{2} F(t_{n+1}, y_{n+1}) - \frac{1}{2} F(t_n, y_n) \right)$$

- 3-ий порядок:

$$y_{n+3} = y_{n+2} + \tau \left( \frac{23}{12} F(t_{n+2}, y_{n+2}) - \frac{4}{3} F(t_{n+1}, y_{n+1}) + \frac{5}{12} F(t_n, y_n) \right)$$

- 4-ый порядок:

$$y_{n+4} = y_{n+3} + \tau \left( \frac{55}{24} F(t_{n+3}, y_{n+3}) - \frac{59}{24} F(t_{n+2}, y_{n+2}) + \frac{37}{24} F(t_{n+1}, y_{n+1}) - \frac{3}{8} F(t_n, y_n) \right)$$

- 5-ый порядок:

$$y_{n+5} = y_{n+4} + \tau \left( \frac{1901}{720} F(t_{n+4}, y_{n+4}) - \frac{1387}{360} F(t_{n+3}, y_{n+3}) + \frac{109}{30} F(t_{n+2}, y_{n+2}) - \frac{637}{360} F(t_{n+1}, y_{n+1}) + \frac{251}{720} F(t_n, y_n) \right)$$

Для нахождения неизвестных начальных значений используется метод Рунге-Кутта 4-го порядка.

### Метод Адамса-Мултона

Неявные методы Адамса-Мултона:

- 1-ый порядок:

$$y_n = y_{n-1} + \tau F(t_n, y_n)$$

- 2-ой порядок:

$$y_{n+1} = y_n + \frac{\tau}{2} (F(t_{n+1}, y_{n+1}) + F(t_n, y_n))$$

- 3-ий порядок:

$$y_{n+2} = y_{n+1} + \tau \left( \frac{5}{12} F(t_{n+2}, y_{n+2}) + \frac{2}{3} F(t_{n+1}, y_{n+1}) - \frac{1}{12} F(t_n, y_n) \right)$$

- 4-ый порядок:

$$y_{n+3} = y_{n+2} + \tau \left( \frac{3}{8} F(t_{n+3}, y_{n+3}) + \frac{19}{24} F(t_{n+2}, y_{n+2}) - \frac{5}{24} F(t_{n+1}, y_{n+1}) + \frac{1}{24} F(t_n, y_n) \right)$$

- 5-ый порядок:

$$y_{n+4} = y_{n+3} + \tau \left( \frac{251}{720} F(t_{n+4}, y_{n+4}) + \frac{646}{720} F(t_{n+3}, y_{n+3}) - \frac{264}{720} F(t_{n+2}, y_{n+2}) + \frac{106}{720} F(t_{n+1}, y_{n+1}) - \frac{19}{720} F(t_n, y_n) \right)$$

Для нахождения неизвестных начальных значений используется метод Рунге-Кутта 4-го порядка.

### Методы Гира

Метод Гира 1-го порядка: $y_n - y_{n-1} = \tau F(t_n, y_n)$

Метод Гира 2-го порядка: $3y_n - 4y_{n-1} + y_{n-2} = 2\tau F(t_n, y_n)$

Метод Гира 3-го порядка: $11y_n - 18y_{n-1} + 9y_{n-2} - 2y_{n-3} = 6\tau F(t_n, y_n)$

Метод Гира 4-го порядка: $25y_n - 48y_{n-1} + 36y_{n-2} - 16y_{n-3} + y_{n-4} = 12\tau F(t_n, y_n)$

Для нахождения неизвестных начальных значений используется метод Рунге-Кутта 4-го порядка.

## 2. Аналитическое решение

Необходимо решить дифференциальное уравнение

$$\frac{\mathrm{d}U}{\mathrm{d}t} = 2sin(t) + U$$

с начальными условиями $U(0) = 5$.

Решение:

$$U(t) = 6e^t - sin(t) - cos(t)$$

## 3. Реализация методов решения ОДУ

Для решения ДУ были использованы: методы Адамса-Башфорта-Мултона 2-го порядка, метод Гира 2-го порядка.

**Реализация данных методов:**

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy import optimize
from typing import Callable
import math
```

In [2]:
```
def du(t: float, u: float) -> float:
    return 2 * math.sin(t) + u


def analytical_du(t: list, i: int) -> list:
    return 6 * math.exp(t[i]) - math.sin(t[i]) - math.cos(t[i])


def runge_kutta(du_: Callable, order: int, tau: float, u: list, t: list) -> list:
    for i in range(order - 1):
        k1 = du_(t[i], u[i])
        k2 = du_(t[i] + tau / 2, u[i] + tau * k1  / 2)
        k3 = du_(t[i] + tau / 2, u[i] + tau * k2  / 2)
        k4 = du_(t[i] + tau, u[i] + tau * k3)
        u[i + 1] = u[i] + tau * (k1 + 2 * k2 + 2 * k3 + k4) / 6
    return u


def analytical_solution(tau: float, T_size: float, u0: int = 5) -> list and list:
    amount_t = int(round(T_size / tau))
    t = np.linspace(0, amount_t * tau, amount_t + 1)
    u = np.zeros((amount_t + 1, len(u0)))
    for i in range(amount_t + 1):
        u[i] = analytical_du(t, i)
    return u, t
```

```python
def adams_bashfort(du: Callable, u0: list, tau: float, T_size: float) -> list and list:
    amount_t = int(round(T_size / tau))
    du_ = lambda t, u: np.asarray(du(t, u))
    t = np.linspace(0, amount_t * tau, amount_t + 1)
    u = np.zeros((amount_t + 1, len(u0)))
    u[0] = u0

    u = runge_kutta(du_=du_, order=2, tau=tau, u=u, t=t)

    for i in range(amount_t - 1):
        u[i + 2] = u[i + 1] + tau * (1.5 * du_(t[i + 1], u[i + 1]) - 0.5 * du_(t[i], u[i]))
    return u, t


def adams_multon(du: Callable, u0: list, tau: float, T_size: float) -> list and list:
    amount_t = int(round(T_size / tau))
    du_ = lambda t, u: np.asarray(du(t, u))
    t = np.linspace(0, amount_t * tau, amount_t + 1)
    u = np.zeros((amount_t + 1, len(u0)))
    u[0] = u0

    def func(a, t, prev_t, b):
        return a - b - tau * (du_(t, a) + du_(prev_t, b)) / 2

    for i in range(amount_t):
        u[i + 1] = optimize.fsolve(func, u[i], (t[i + 1], t[i], u[i]))
    return u, t


def gear_second(du: Callable, u0: list, tau: float, T_size: float) -> list and list:
    amount_t = int(round(T_size / tau))
    du_ = lambda t, u: np.asarray(du(t, u))
    t = np.linspace(0, amount_t * tau, amount_t + 1)
    u = np.zeros((amount_t + 1, len(u0)))
    u[0] = u0

    u = runge_kutta(du_=du_, order=2, tau=tau, u=u, t=t)

    def func(a, t, b, c):
        return 3 * a - 4 * b + c - 2 * tau * du_(t, a)

    for i in range(amount_t - 1):
        u[i + 2] = optimize.fsolve(func, u[i + 1], (t[i + 2], u[i + 1], u[i]))
    return u, t


T_size = 1
```

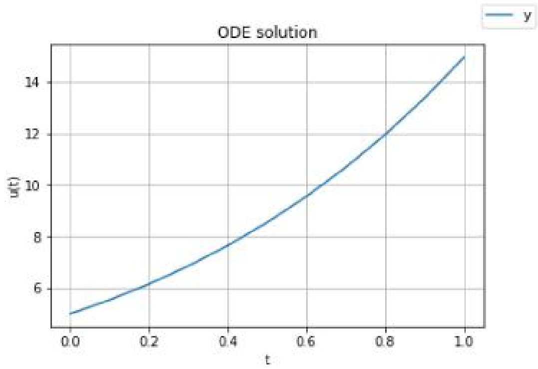
**Функции отрисовки:**

In [3]:
```python
def draw(t: list, u: list):
    fig = plt.figure()
    plt.title('ODE solution')
    plt.ylabel('u(t)')
    plt.xlabel('t')
    l1 = plt.plot(t, u)
    fig.legend((l1), ('y'))
    plt.grid(True)
    plt.show()


def draw_analytical_solution():
    u, t = analytical_solution(tau=tau, T_size=T_size, u0=[5])
    draw(t, u)


def draw_adams_bashfort():
    u, t = adams_bashfort(du, u0=[5], tau=tau, T_size=T_size)
    draw(t, u)


def draw_adams_multon():
    u, t = adams_multon(du, u0=[5], tau=tau, T_size=T_size)
    draw(t, u)


def draw_gear_second():
    u, t = gear_second(du, u0=[5], tau=tau, T_size=T_size)
    draw(t, u)
```

## 4. Итоговые результаты

### Аналитическое решение

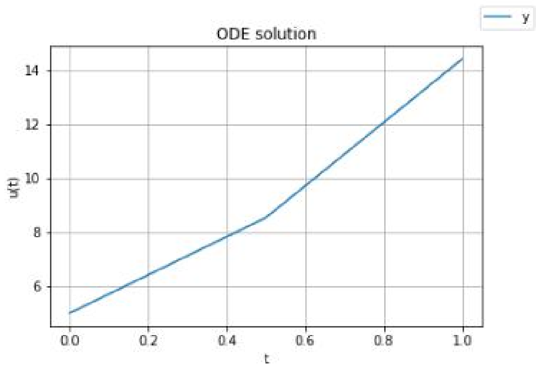
In [4]:
```
tau = 0.1
draw_analytical_solution()
```




### Метод Адамса-Башфорта

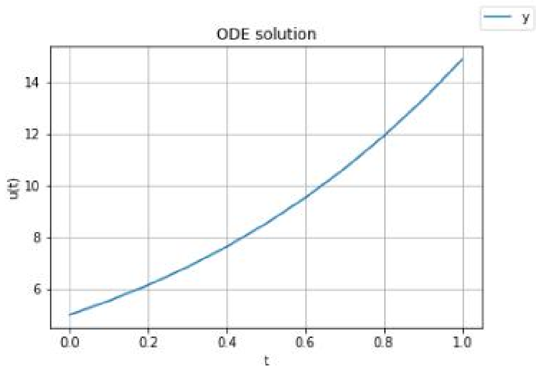
Шаг $\tau = 0.5$

In [5]:
```
tau = 0.5
draw_adams_bashfort()
```




Шаг $\tau = 0.1$

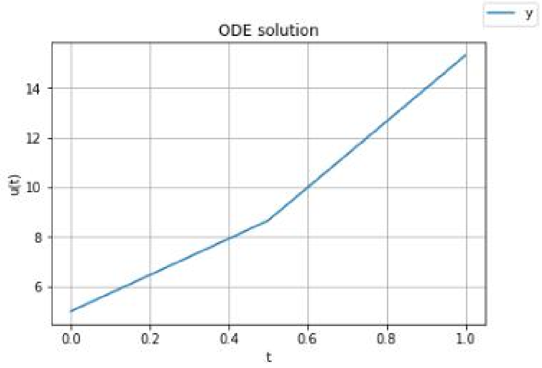
In [6]:
```
tau = 0.1
draw_adams_bashfort()
```



### Метод Адамса-Мултона

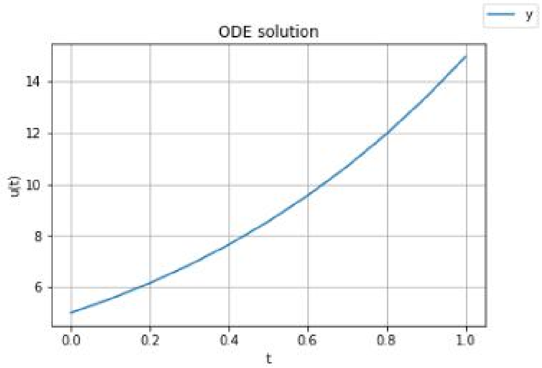
Шаг $\tau = 0.5$

In [7]:
```
tau = 0.5
draw_adams_multon()
```




Шаг $\tau = 0.1$

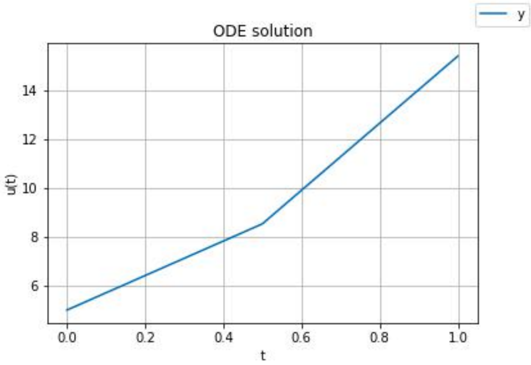
In [8]:
```
tau = 0.1
draw_adams_multon()
```



### Метод Гира 2-го порядка

Шаг $\tau$ = 0.5

In [9]:
```
tau = 0.5
draw_gear_second()
```




Шаг $\tau$ = 0.1

In [10]:
```
tau = 0.1
draw_gear_second()
```