Университет ИТМО МФ КТиУ, Ф ПИиКТ

**Лабораторная работа №6**

**Дисциплина «Вычислительная математика»**

**Численное решение обыкновенных дифференциальных уравнений**

**Выполнил** Аскаров Эмиль Рамилевич

**Преподаватель:**

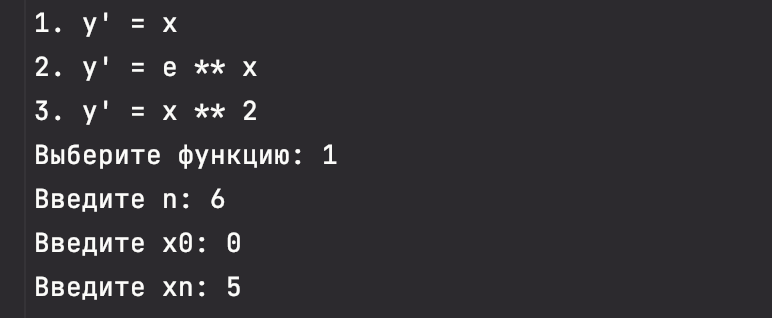
Машина Екатерина Алексеевна

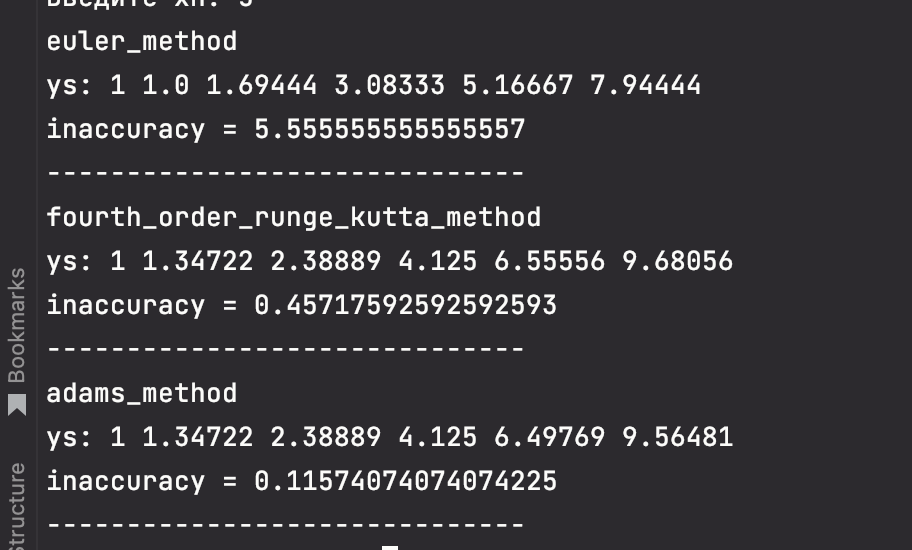
г. Санкт-Петербург 2024 г.

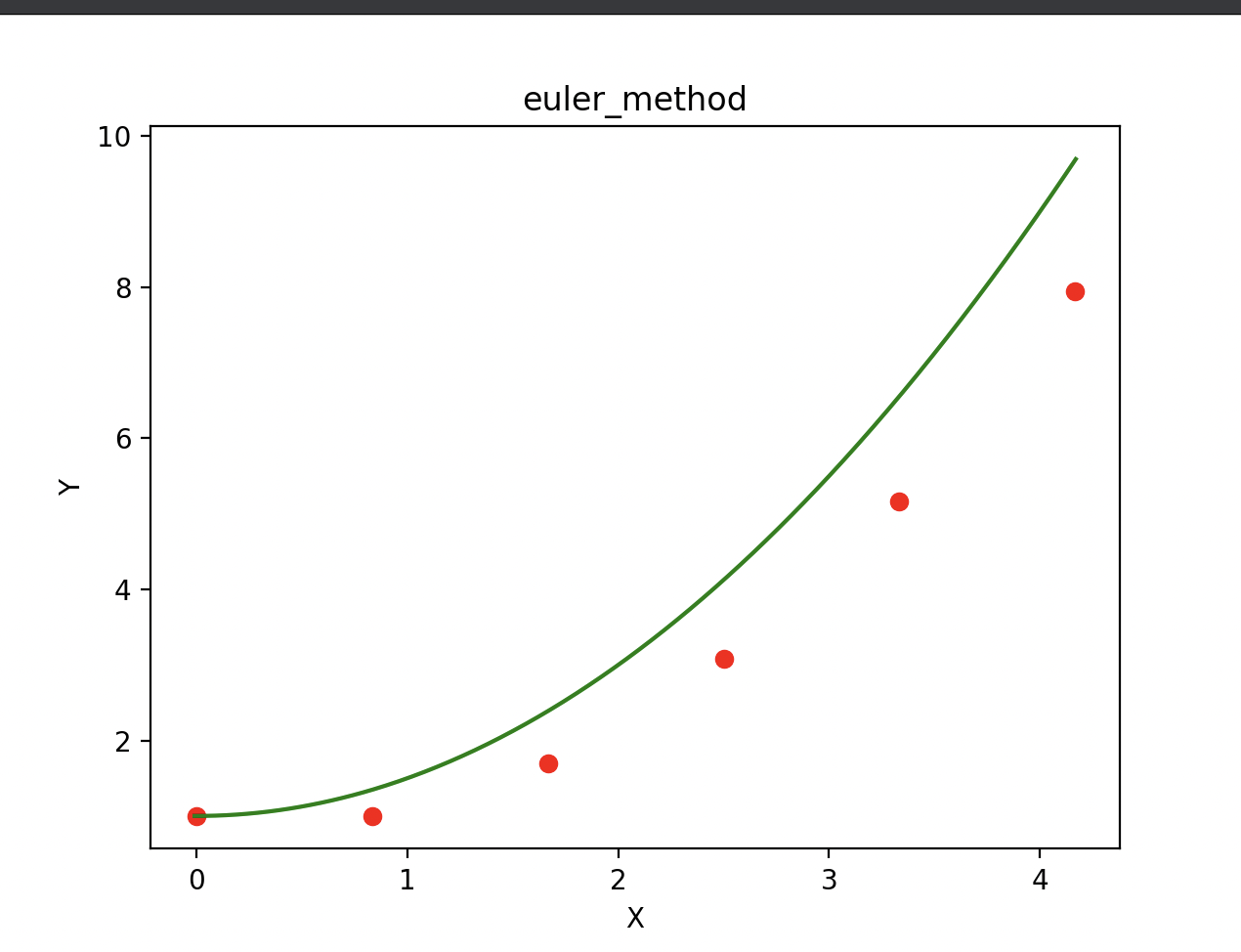
# Программная реализация задачи

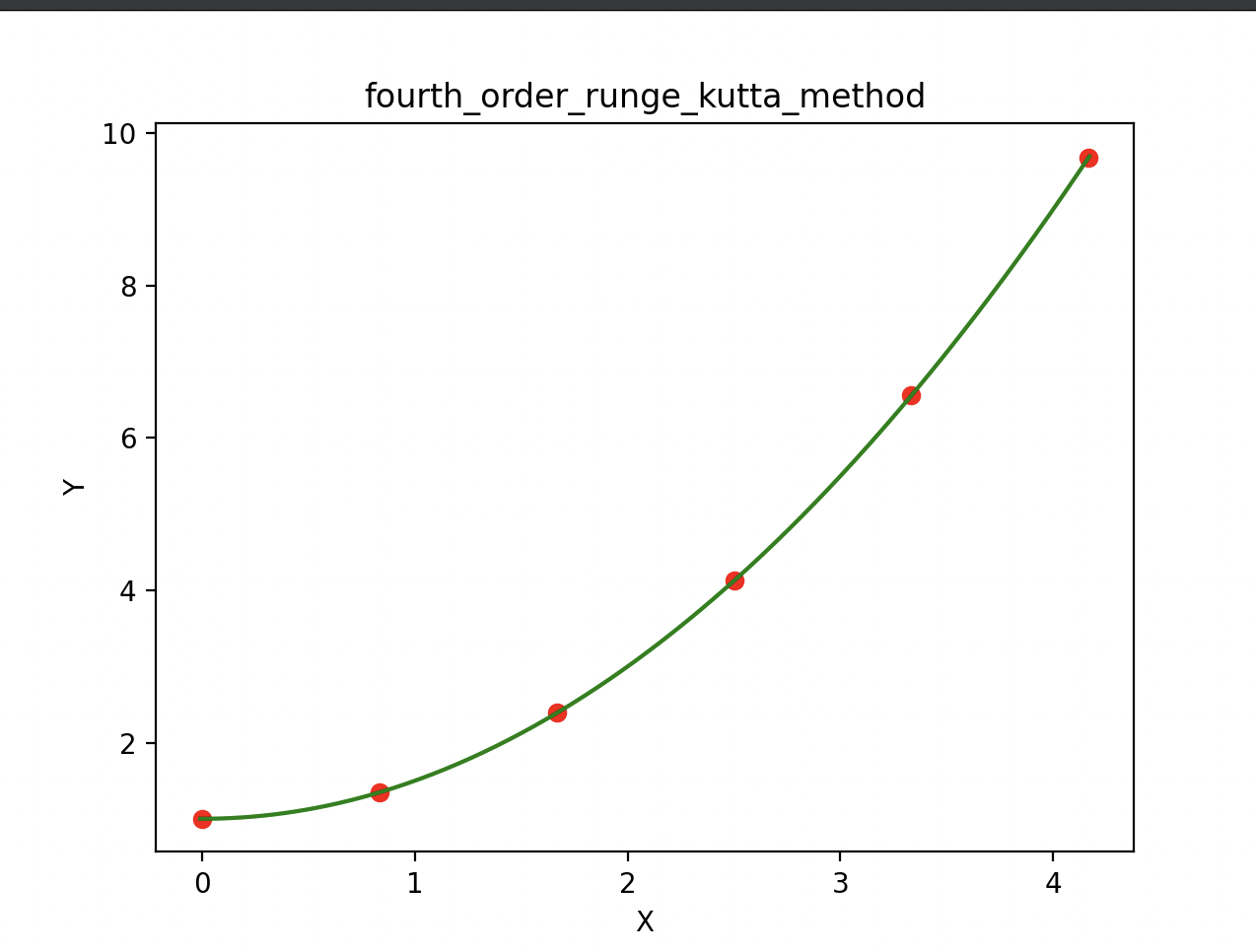
from matplotlib import pyplot as plt  
from math import exp  
  
  
def euler\_method(*f*, *xs*, *y0*):  
 ys = [*y0*]  
 h = *xs*[1] - *xs*[0]  
 for i in *range*(1, *len*(*xs*)):  
 ys.append(ys[i - 1] + h \* *f*(*xs*[i - 1], ys[i - 1]))  
 return ys  
  
  
def fourth\_order\_runge\_kutta\_method(*f*, *xs*, *y0*):  
 ys = [*y0*]  
 h = *xs*[1] - *xs*[0]  
 for i in *range*(1, *len*(*xs*)):  
 k1 = h \* *f*(*xs*[i - 1], ys[i - 1])  
 k2 = h \* *f*(*xs*[i - 1] + h / 2, ys[i - 1] + k1 / 2)  
 k3 = h \* f(*xs*[i - 1] + h / 2, ys[i - 1] + k2 / 2)  
 k4 = h \* *f*(*xs*[i - 1] + h, ys[i - 1] + k3)  
 ys.append(ys[i - 1] + 1 / 6 \* (k1 + 2 \* k2 + 2 \* k3 + k4))  
 return ys  
  
  
def adams\_method(*f*, *xs*, *y0*):  
 ys = fourth\_order\_runge\_kutta\_method(*f*, *xs*[:4], *y0*)  
 h = *xs*[1] - *xs*[0]  
  
 for i in *range*(4, *len*(*xs*)):  
 df = *f*(*xs*[i - 1], ys[i - 1]) - *f*(*xs*[i - 2], ys[i - 2])  
 d2f = *f*(*xs*[i - 1], ys[i - 1]) - 2 \* *f*(*xs*[i - 2], ys[i - 2]) + *f*(*xs*[i - 3], ys[i - 3])  
 d3f = *f*(*xs*[i - 1], ys[i - 1]) - 3 \* *f*(*xs*[i - 2], ys[i - 2]) + 3 \* *f*(*xs*[i - 3], ys[i - 3]) - *f*(*xs*[i - 4], ys[i - 4])  
 y = ys[i - 1] + h \* f(*xs*[i - 1], ys[i - 1]) + h \*\* 2 / 2 \* df + 5 \* h \*\* 3 / 12 \* d2f + 3 \* h \*\* 4 / 8 \* d3f  
 ys.append(y)  
 return ys  
  
  
def draw\_plot(*a*, *b*, *func*, *dx*=0.01):  
 xs, ys = [], []  
 *a* -= *dx  
 b* += *dx* x = *a* while x <= *b*:  
 xs.append(x)  
 ys.append(*func*(x))  
 x += *dx* plt.plot(xs, ys, 'g')  
  
  
def main(*f*, *xs*, *y0*, *exact\_y*):  
 methods = [euler\_method,  
 fourth\_order\_runge\_kutta\_method,  
 adams\_method]  
 for method in methods:  
 *print*(method.*\_\_name\_\_*)  
  
 ys = method(*f*, *xs*, *y0*)  
 if method in (adams\_method,):  
 inaccuracy = *max*([*abs*(exact\_y(x) - y)  
 for x, y in *zip*(*xs*, ys)])  
 else:  
 xs2 = []  
 for x1, x2 in *zip*(*xs*, *xs*[1:]):  
 xs2.extend([x1, (x1 + x2) / 2, x2])  
 ys2 = method(*f*, xs2, *y0*)  
 p = 4 if method is fourth\_order\_runge\_kutta\_method else 1  
 inaccuracy = *max*([*abs*(y1 - y2) / (2 \*\* p - 1) for y1, y2 in *zip*(ys, ys2)])  
 *print*("ys:", \**map*(lambda *x*: *round*(x, 5), ys))  
 *print*(f"inaccuracy = {inaccuracy}")  
  
 plt.title(method.*\_\_name\_\_*)  
  
 draw\_plot(*xs*[0], *xs*[-1], *exact\_y*)  
 for i in *range*(*len*(*xs*)):  
 plt.scatter(*xs*[i], ys[i], *c*='r')  
 plt.xlabel("X")  
 plt.ylabel("Y")  
 plt.show()  
 *print*('-' \* 30)  
  
  
def read\_number(*s*: *str*):  
 while True:  
 try:  
 return *float*(*input*(*s*))  
 except *Exception*:  
 continue  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 *print*("1. y' = x")  
 *print*("2. y' = e \*\* x")  
 *print*("3. y' = x \*\* 2")  
 mode = read\_number("Выберите функцию: ")  
 n = read\_number("Введите n: ")  
 x0 = read\_number("Введите x0: ")  
 xn = read\_number("Введите xn: ")  
 h = (xn - x0) / n  
 xs = [x0 + h \* i for i in *range*(*int*(n))]  
 try:  
 if mode == 1:  
 *# xs = [0, 1, 2, 3, 4, 5]* f = lambda x, y: x  
 y0 = 1  
 exact\_y = lambda x: x \*\* 2 / 2 + 1  
 elif mode == 2:  
 *# xs = [0, 1, 2, 3, 4, 5]* f = lambda x, y: exp(x)  
 y0 = 0  
 exact\_y = lambda x: exp(x) - 1  
 elif mode == 3:  
 *# xs = [0, 1, 2, 3, 4, 5]* f = lambda x, y: x \*\* 2  
 y0 = 5  
 exact\_y = lambda x: x \*\* 3 / 3 + 5  
 except (*ZeroDivisionError*, *ArithmeticError*) as e:  
 *print*("Функция не определена")  
  
 main(f, xs, y0, exact\_y)

# Тестовые данные









# Вывод

В ходе лабораторной работы я познакомился с численным решением обыкновенных дифференциальных уравнений.

Метод Эйлера – простой, но неточный метод, одношаговый.

Модификация метода Эйлера – более точный чем оригинал.

Методы Рунге-Кутта – хороший метод, но требует много вычислений по сравнению с прошлыми двумя, одношаговый.

Метод Адамса – многошаговый метод, точный. Использует на каждом шаге результаты предыдущих четырёх шагов. Использует конечные разности.

Метод Милна – многошаговый метод прогноза и коррекции. Коррекция проводится до тех пор, пока она не будет похожа на прогноз. Тоже использует результаты предыдущих четырёх шагов.