МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО «МГТУ «СТАНКИН»)

Институт информационных систем и технологий	Кафедра прикладной математики

ОТЧЕТ О ВЫПОЛНЕНИИ
ЛАБОРАТОРНОЙ РАБОТЫ № 3
ПО ДИСЦИПЛИНЕ «ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА»
НА ТЕМУ «ВСПЛЫТИЕ ПОДВОДНОЙ ЛОДКИ»

СТУДЕНТА 2 КУРСА бакалавриата ГРУППЫ ИДБ-22-15

(уровень профессионального образования)

Набойщикова Артемия Андреевича

(Фамилия Имя Отчество)

Направление: 09.03.04 «Программная инженерия»

Профиль подготовки: «Системный анализ и проектирование программных комплексов»

Отчет сдан: «3» июня 2024 г.

Проверил: Москалев П.В., профессор, д.ф.-м.н.

(Фамилия И.О. должность/звание, степень) *(Подпись)*

МОСКВА 2024

Лабораторная работа № 3: «Всплытие подводной лодки»

Цель работы: изучить методы численного дифференцирования для решения системы обыкновенных дифференциальных уравнений и применить их на практике для решения прикладной задачи (построения траектории, определения времени и точки всплытия подводной лодки).

Вывод системы обыкновенных дифференциальных уравнений

По оси абсцисс $\frac{dx}{dt} = v, x = vt$

Тогда: $L = vT$

По второму закону Ньютона получим:

$$m\frac{d^2y}{dt^2} = F_B - P - F_c$$

$$m = \rho_1 V, F_B = \rho_0 V g,$$

$$P = \rho_1 V g,$$

$$F_c = k\eta\frac{dy}{dt}, k = \frac{S_{\text{сеч}}}{l}$$

Подставим и получим:

$$\rho_1 V\frac{d^2y}{dt^2} = \rho_0 V g - \rho_1 V g - k\eta(1 + \alpha\frac{y}{H})\frac{dy}{dt} \qquad (1)$$

Здесь: $\eta = 0{,}001$; $\rho_0 = 1000$; $g = 9{,}8$; $\alpha = 0{,}01$

Самим задать $V, S_{\text{сеч}}, l, H, \rho_1, v$

Получим систему:

$$\begin{cases} \frac{dy}{dt} = z \\ \frac{dz}{dt} = -\frac{\eta k}{V\rho_1}\left(1 + \alpha\frac{y}{H}\right)z + g(\frac{\rho_0}{\rho_1} - 1) \end{cases} \qquad (2)$$

Постановка задачи. Дано: *H* — глубина погружения подводной лодки. Найти: *T* — время всплытия подводной лодки; *L* — абсцисса точки всплытия подводной лодки.

Задание на лабораторную работу

1. Численно решить систему дифференциальных уравнений, используя явные методы Эйлера–Коши второго порядка или Рунге–Кутты четвертого порядка. Оценить погрешность по правилу Рунге. Шаг по глубине выбирать в пределах от 0,01 *H* до 0,001 *H*.
2. Аппроксимировать полученное решение по методу наименьших квадратов, используя не менее 20 точек. Для аппроксимации использовать полином второго порядка вида

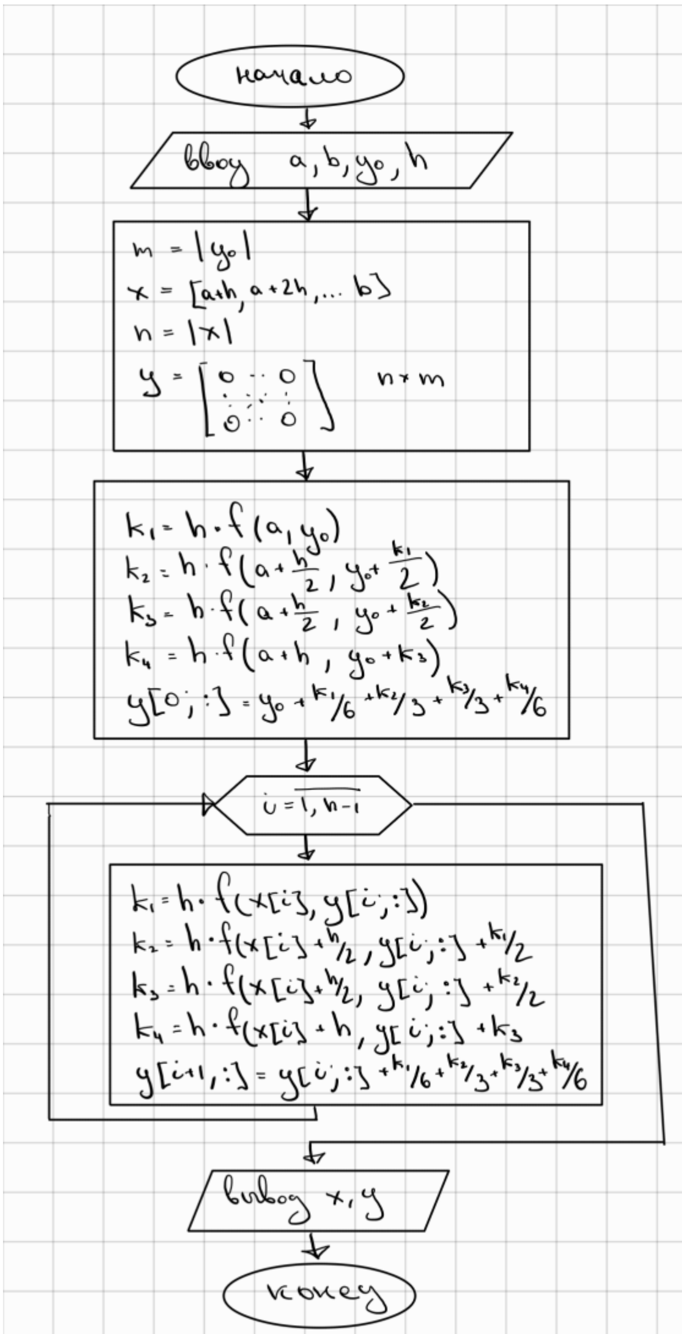
$y = b_0 + b_1 t + b_2 t^2$. Для оценки погрешности аппроксимации использовать среднее квадратичное отклонение.

3. По полученным точкам построить траекторию всплытия подводной лодки $y = f(x)$.
4. Используя построенную аппроксимацию $b_0 + b_1 T + b_2 T^2 = 0$, оценить значения времени всплытия T и абсциссы точки всплытия подводной лодки $L = vT$.
5. Провести анализ выполненной лабораторной работы и сделать выводы.

Выполнение лабораторной работы

1. Численное решение системы обыкновенных дифференциальных уравнений

Блок-схема явного метода Рунге–Кутты четвертого порядка



Листинг реализации явного метода Рунге–Кутты четвертого порядка для решения системы обыкновенных дифференциальных уравнений на языке Python

```python
from typing import Callable, List, Tuple
import numpy as np
import matplotlib.pyplot as plt

# Дано
eta = 0.001
rho0 = 1025
g = 9.8
alpha = 0.01
rho1 = 950

# Задать самому
V = 900 # объём
S = 180 # площадь
l = V/S # длина
H = 50 # глубина
v = 10 # скорость

k = S / l

# Метод Рунге-Кутта 4 порядка
def rk4(f: Callable[[float, np.ndarray], np.ndarray], a: float, b: float,
y0: np.ndarray, h: int) -> Tuple[np.ndarray, np.ndarray]:
    m = len(y0)
    x = np.arange(a + h, b, h)
    n = len(x)
    y = np.zeros((n, m))

    k1 = h * f(a, y0)
    k2 = h * f(a + h/2, y0 + k1/2)
    k3 = h * f(a + h/2, y0 + k2/2)
    k4 = h * f(a + h, y0 + k3)

    y[0, :] = y0 + k1/6 + k2/3 + k3/3 + k4/6

    for i in range(n - 1):
        k1 = h * f(x[i], y[i, :])
        k2 = h * f(x[i] + h/2, y[i, :] + k1/2)
        k3 = h * f(x[i] + h/2, y[i, :] + k2/2)
        k4 = h * f(x[i] + h, y[i, :] + k3)
        y[i + 1, :] = y[i, :] + k1/6 + k2/3 + k3/3 + k4/6
    return x, y


# Функция для системы уравнений
def submarine_sys(t: float, y: np.ndarray):
```

```
    dy = y[1]
    dz = -(eta*k / (V*rho1)) * (1 + alpha*y[0]/H)*dy + g*(rho0/rho1-1)
    return np.array([dy, dz])

# Начальные условия
y0 = np.array([0, 0])
a = 0
b = 15
h = 0.01

# Решение системы дифференциальных уравнений
solution: Tuple[np.ndarray, np.ndarray] = rk4(submarine_sys, a, b, y0, h)

solution_x = solution[0][solution[1][:, 0] <= H]
solution_y = solution[1][solution[1][:, 0] <= H]


# Расчёт погрешности по правилу Рунге
half =  rk4(submarine_sys, a, b, y0, h / 2)
half_x = half[0][half[1][:, 0] <= H]
half_y = half[1][half[1][:, 0] <= H]

runge_errs = [solution_y[:, 0][i + 1] - half_y[:, 0][i * 2] for i in
range(len(solution_y[:,0]) - 1)]
runge_err = max(runge_errs) / 15
print(f"Погрешность по правилу Рунге: {runge_err}")
```

Вычислим погрешность по правилу Рунге:

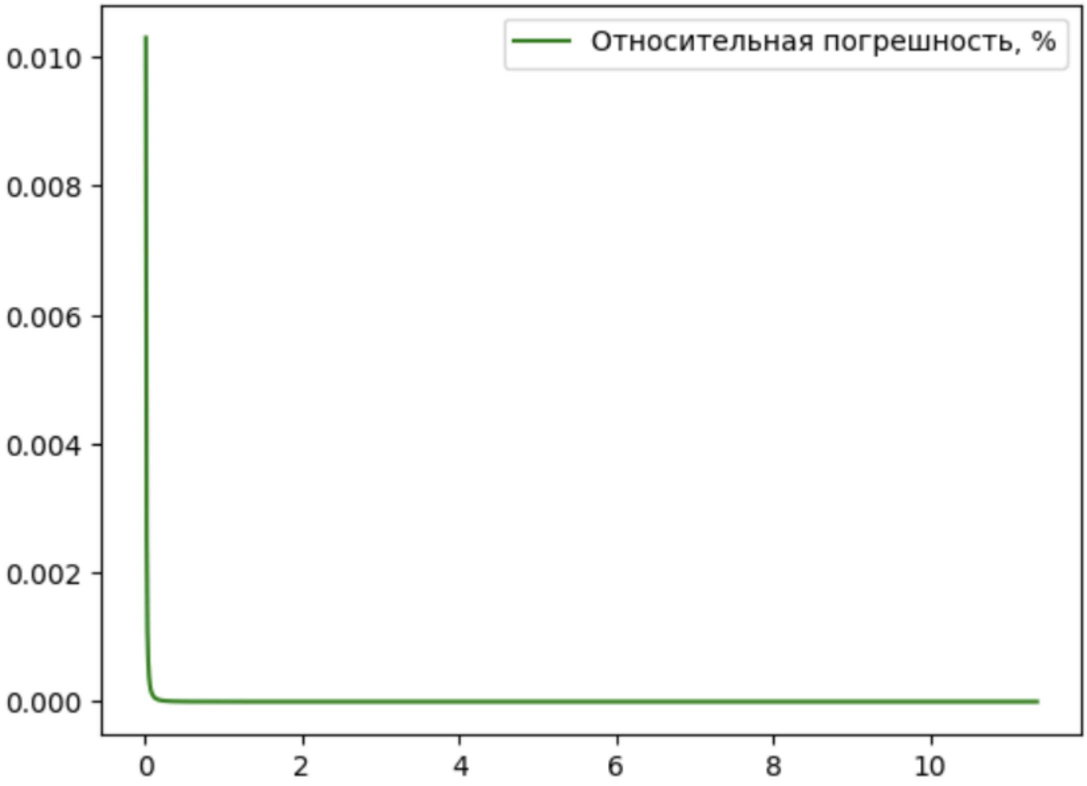
$$d_i(h/2) = \frac{\left|y_i(h) - y_i\left(\frac{h}{2}\right)\right|}{2^p - 1} \quad D = \max\left(d_i\left(\frac{h}{2}\right)\right) = 0{,}008783247890340344$$

2. Квадратичная аппроксимация решения по методу наименьших квадратов

```
# Функция для аппроксимации методом наименьших квадратов
def gaussian_elimination(matrix: np.ndarray, vector: np.array) -> np.ndar-
ray:
    n = len(vector)

    # Прямой ход метода Гаусса
    for i in range(n-1):
        pivot = matrix[i, i]
        for k in range(i+1, n):
            factor = matrix[k, i] / pivot
            matrix[k, i:] -= factor * matrix[i, i:]
            vector[k] -= factor * vector[i]
```

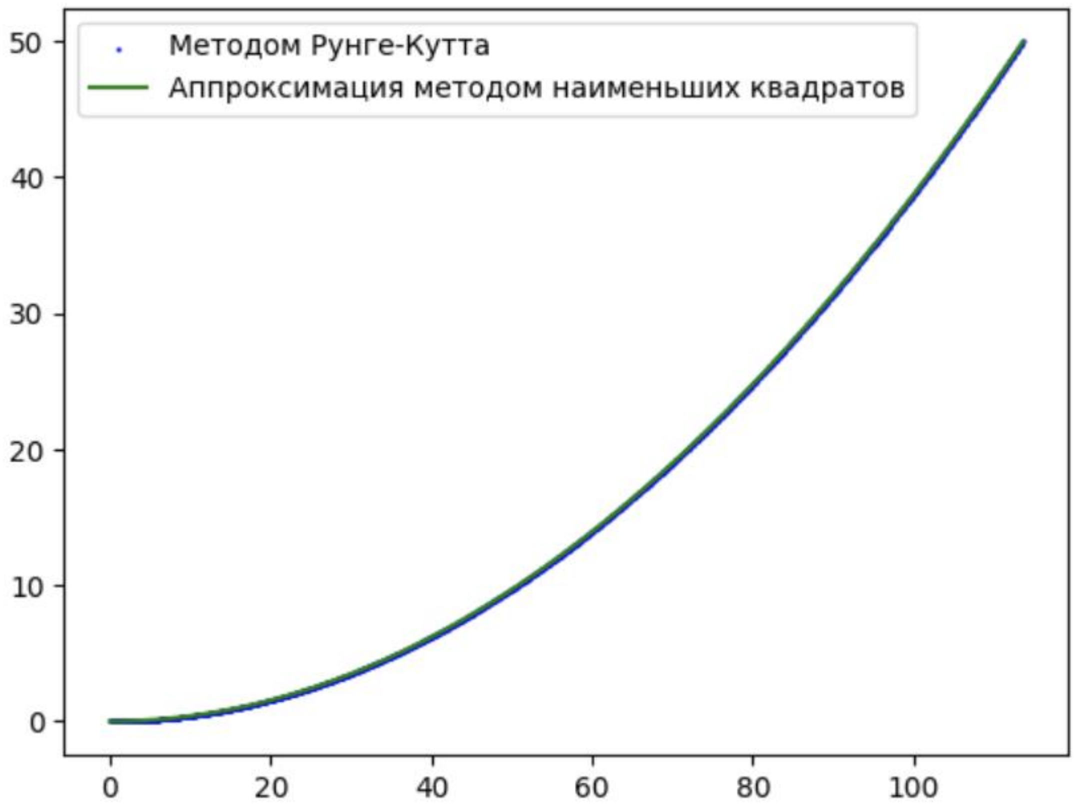
```
    # Обратный ход метода Гаусса
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (vector[i] - np.dot(matrix[i, i+1:], x[i+1:])) / matrix[i, i]

    return x

def approx(x: np.ndarray, y: np.array, degree: int) -> np.ndarray:
    n: int = len(x)
    X: np.ndarray = np.ones((n, degree + 1))
    for i in range(1, degree + 1):
        X[:, i] = x ** i
    coefficients: np.ndarray = gaussian_elimination(np.dot(X.T, X),
np.dot(X.T, y))
    return coefficients

# Аппроксимация решения и получение коэффициентов полинома
degree = 2
c,b,a = approx(solution_x, solution_y[:, 0], degree)
print("Коэффициенты квадратичного полинома методом наименьших квадратов: ")
print(f"a = {a}")
print(f"b = {b}")
print(f"c = {c}")
def f_approx(t):
    return a * t**2 + b*t + c

# Вычисление среднеквадратичного отклонения
t_approx = list(solution_x)
y_approx = [f_approx(t) for t in t_approx]
abs_errs = [abs(y_approx[i] - solution_y[:, 0][i]) for i in range(len(solu-
tion_y[:,0]))]
rel_errs = [abs(abs_errs[i] / solution_y[:,0][i]) for i in
range(len(abs_errs))]
std_err = sum([e**2 for e in abs_errs]) / len(abs_errs)
print(f'Среднеквадратическоеное отклонение: {std_err}')

print(rel_errs)
plt.plot(t_approx, rel_errs, color="green", label="Относительная погреш-
ность, %" )
plt.legend()
plt.show()
```

```
Коэффициенты квадратичного полинома методом наименьших квадратов:
a = 0.38684201233479115
b = 4.2357607454004024e-07
c = -4.026487370627903e-07
Среднеквадратичное отклонение: 2.30045983564933e-14
```



3. Траектория всплытия подводной лодки

```
# Построение графика аппроксимации функции
t_plot = np.linspace(0, np.max(solution_x ), 100)
y_plot = [f_approx(t) for t in t_plot]
plt.scatter(solution_x * v, solution_y[:, 0], color="blue", marker="o",
s=0.5, label="Методом Рунге-Кутта")
plt.plot(t_plot * v, y_plot, color="green", label="Аппроксимация методом
наименьших квадратов" )
plt.legend()
plt.show()
```



4. Оценка значений времени всплытия T и абсциссы точки всплытия подводной лодки L

```
# Определение времени всплытия и точки всплытия
def get_time(a: float, b: float, c: float, H: float) -> float:
    roots = np.roots([a, b, c - H])
    return roots[roots > 0]

T = get_time(a, b, c, H)
L = v * T

# Вывод значений
print("Время всплытия (T):", T)
print("Абсцисса точки всплытия (L):", L)
```

```
Время всплытия (T): [11.36889266]
Абсцисса точки всплытия (L): [113.6889266]
```

5. Анализ выполненной работы и выводы

Изучили метод (метод Рунге–Кутты четвертого порядка) численного дифференцирования для решения системы обыкновенных дифференциальных уравнений и применили их на практике для решения прикладной задачи (построения траектории, определения времени и точки всплытия подводной лодки).