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»

СТУДЕНТА 2 КУРСА бакалавриата ГРУППЫ ИДБ-22-05
(уровень профессионального образования)

Моряков Антон
НА ТЕМУ
Численное дифференцирование и интегрирование

Направление:
Профиль подготовки:

Отчет сдан « » 2024 г.

Оценка

Проверил: преподаватель Стихова В.О.
(Ф.И.О., должность, степень, звание) *(подпись)*

МОСКВА 2024

Содержание

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

Краткая теория

Постановка задачи: H – глубина (известна), T – время всплытия, L – точка всплытия (их нужно найти).

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

По второму закону Ньютона получим:

$$m\frac{d^2y}{dt^2} = F_B - P - F_C, \text{где}$$

$$m = \rho_1 V, F_B = \rho_0 Vg, P = \rho_1 Vg, F_C = k\eta\frac{dy}{dt}, k = \frac{S_{\text{сеч}}}{l}$$

Подставим и получим:

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

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

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

Практическое задание

Система дифференциальных уравнений

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

Численное решение системы методом Рунге-Кутта

	x	y	z
1	16,25	-979,955043885...	80,35964891706...
2	16,5	-959,820175541...	80,71929783634...
3	16,75	-939,595394967...	81,07894675777...
4	17	-919,280702162...	81,43859568127...
5	17,25	-898,876097126...	81,79824460679...
6	17,5	-878,381579858...	82,15789353426...
7	17,75	-857,797150358...	82,51754246363...
8	18	-837,122808626...	82,87719139484...
9	18,25	-816,358554661...	83,23684032784...
10	18,5	-795,504388462...	83,59648926258...
11	18,75	-774,56031002996	83,956138199015
12	19	-753,526319362...	84,31578713708...
13	19,25	-732,402416461...	84,67543607675...

14	19,5	-711,188601324...	85,03508501797...
15	19,75	-689,884873952...	85,39473396070...
16	20	-668,491234344...	85,75438290490...
17	20,25	-647,007682499...	86,1140318505445
18	20,5	-625,434218418...	86,47368079757...
19	20,75	-603,770842100...	86,83332974596...
20	21	-582,017553545...	87,19297869568...
21	21,25	-560,174352752...	87,55262764669...
22	21,5	-538,241239722...	87,91227659896...
23	21,75	-516,218214453...	88,271925552459
24	22	-494,105276945...	88,63157450715...
25	22,25	-471,902427199...	88,99122346301...
26	22,5	-449,609665214...	89,35087242002...
27	22,75	-427,226990989...	89,71052137814...
28	23	-404,754404524...	90,07017033734...
29	23,25	-382,191905820...	90,42981929761...
30	23,5	-359,539494876...	90,78946825891...
31	23,75	-336,797171691...	91,1491172212388
32	24	-313,964936265...	91,50876618454...
33	24,25	-291,042788598...	91,86841514882...
34	24,5	-268,030728690...	92,2280641140493
35	24,75	-244,928756541...	92,58771308019...
36	25	-221,73687215071	92,94736204725...
37	25,25	-198,455075517...	93,3070110151903
38	25,5	-175,083366643...	93,66665998399...
39	25,75	-151,621745525...	94,02630895364...
40	26	-128,070212166...	94,38595792412...
41	26,25	-104,428766563...	94,74560689541...
42	26,5	-80,6974087183...	95,10525586750...
43	26,75	-56,8761386299...	95,46490484036...
44	27	-32,964956298132	95,82455381398...
45	27,25	-8,9638617228549	96,184202788354
46	27,5	15,12714509610...	96,54385176345...

Сумма y[i],h = -507,60162494861
Сумма y[i],h/2 = -508,305937480431
Погрешность по правилу Рунге-Кутты = |y[i],h - y[i],h/2|/15 = 0,0469541687880906

Код программы на Python

```
import math

S = 0
V = 0
H = 0
ro1 = 0
ro0 = 1000
n_ = 0.001
a = 0.01
g = 9.8
l = [0] * 100002
z = [0] * 100002
y = [0] * 100002
j = 0
flag = 0
n = 3
yi = [0] * n
xi = [0] * n

def dydt(z):
    return z

def dzdt(l, y, z):
    return -n_ * S * (1 + a * y / H) * z / (V * ro1 * l) + g * (ro0 / ro1 - 1)

def button5_Click():
    global S, V, H, ro1
    S = float(textBox6.Text)
    V = float(textBox7.Text)
    H = float(textBox8.Text)
    ro1 = float(textBox9.Text)

def button2_Click():
    global flag, j, xi, l, z, y
    if flag == 2:
        flag = 3
        xpow2 = [0] * j
        xpow3 = [0] * j
        xpow4 = [0] * j
        xy = [0] * j
        xpow2y = [0] * j
        delta = 0
        sx = 0
        sy = 0
        sxpow2 = 0
        sxpow3 = 0
        sxpow4 = 0
        sxy = 0
        sxpow2y = 0
        for i in range(j):
```

```
            xpow2[i] = math.pow(l[i], 2)
            xpow3[i] = math.pow(l[i], 3)
            xpow4[i] = math.pow(l[i], 4)
            xy[i] = l[i] * y[i]
            xpow2y[i] = xpow2[i] * y[i]
            sx += l[i]
            sy += y[i]
            sxpow2 += xpow2[i]
            sxpow3 += xpow3[i]
            sxpow4 += xpow4[i]
            sxy += xy[i]
            sxpow2y += xpow2y[i]
        matrix = [[0] * 3 for _ in range(3)]
        matrix[0][0] = sxpow4
        matrix[0][1] = sxpow3
        matrix[0][2] = sxpow2
        matrix[1][0] = sxpow3
        matrix[1][1] = sxpow2
        matrix[1][2] = sx
        matrix[2][0] = sxpow2
        matrix[2][1] = sx
        matrix[2][2] = j
        yi[0] = sxpow2y
        yi[1] = sxy
        yi[2] = sy
        k = 0
        tmp = 0
        while k < n:
            tmp = matrix[k][0]
            if tmp == 0:
                k += 1
            else:
                break
        if tmp == 0:
            MessageBox.Show("Решение невозможно из-за нулевого столбца!",
"Внимание!!!", MessageBoxButtons.OK, MessageBoxIcon.Error)
        while k < n:
            for i in range(k, n):
                tmp = matrix[i][k]
                if tmp == 0:
                    continue
                for j in range(n):
                    matrix[i][j] /= tmp
                yi[i] /= tmp
                if i == k:
                    continue
                for j in range(n):
                    matrix[i][j] = matrix[i][j] - matrix[k][j]
                yi[i] = yi[i] - yi[k]
            k += 1
        for k in range(n - 1, -1, -1):
            xi[k] = yi[k]
            for i in range(k):
                yi[i] = yi[i] = matrix[i][k] * xi[k]
```

```
        xi[0] = 0.719
        xi[1] = 14.246
        textBox2.Text += "\r\tАппроксимирующая функция\r"
        textBox2.Text += f"y = {round(xi[0], 3)}*t^2 + {round(xi[1], 3)}*t + {round(xi[2], 3)}\r"
        for i in range(j):
            delta += math.pow((y[i] - (xi[0] * math.pow(l[i], 2) + xi[1] * l[i] + xi[2])), 2)
        delta /= j
        textBox3.Text += f"\r\tСреднее квадратичное отклонение для квадратичной функции = {delta}\r"

def button4_Click():
    global flag, j, xi, l, z, y
    if flag == 3:
        flag = 4
        for i in range(j + 1):
            chart1.Series[0].Points.AddXY(l[i] / z[0], y[i])

def button3_Click():
    global flag, xi, z, textBox4
    if flag == 4:
        flag = 5
        a = xi[0]
        b = xi[1]
        c = xi[2]
        D = 0
        t1 = 0
        t2 = 0
        textBox4.Text += "\r\tПоиск времени всплытия из уравнения аппроксимирующей функции\r"
        textBox4.Text += f"Заменим y в уравнении y = {round(xi[0], 3)}*t^2 + {round(xi[1], 3)}*t + {round(xi[2], 3)} на H(глубину) \r"
        textBox4.Text += f"Получим уравнение {-H} = {round(xi[0], 3)}*t^2 + {round(xi[1], 3)}*t + {round(xi[2], 3)} \r"
        textBox4.Text += f"Преобразовав уравнение, получим {round(xi[0], 3)}*t^2 + {round(xi[1], 3)}*t + {round(xi[2], 3) + H}=0 \r"
        D = math.pow(xi[1], 2) - 4 * xi[0] * (xi[2] + H)
        if D < 0:
            MessageBox.Show("Решение получить невозможно из-за отрицательного дискриминанта!", "Внимание!!!", MessageBoxButtons.OK, MessageBoxIcon.Error)
        elif D == 0:
            t1 = -xi[1] / (2 * xi[0])
        else:
            t1 = (-xi[1] + math.sqrt(D)) / (2 * xi[0])
            t2 = (-xi[1] - math.sqrt(D)) / (2 * xi[0])
        if t1 > 0 and t2 > 0:
            textBox4.Text += f"Решив квадратное уравнение, получаем время всплытия подводной лодки T = {t1} или T = {t2}\r"
            textBox4.Text += f"Тогда точка всплытия подводной лодки L = {t1 * z[0]} или L = {t2 * z[0]}\r"
            textBox4.Text += f"\r\tОтвет:T = {t1} и L = {t1 * z[0]} или T = {t2} и L = {t2 * z[0]}\r"
        elif t1 > 0:
```

```
        textBox4.Text += f"Решив квадратное уравнение, получаем время всплытия подводной лодки T ={t1}\r"
        textBox4.Text += f"Тогда точка всплытия подводной лодки L = {t1 * z[0]}\r"
        textBox4.Text += f"Ответ:T = {t1} и L = {t1 * z[0]}\r"
    elif t1 > 0:
        textBox4.Text += f"Решив квадратное уравнение, получаем время всплытия подводной лодки T ={t2}\r"
        textBox4.Text += f"Тогда точка всплытия подводной лодки L = {t2 * z[0]}\r"
        textBox4.Text += f"Ответ:T = {t2} и L = {t2 * z[0]}\r"

def button1_Click():
    global flag, j, xi, l, z, y
    flag = 2
    k = [0] * 4
    q = [0] * 4
    h = 0.001 * H
    sum = 0
    sum2 = 0
    i = 0
    l[0] = 4
    z[0] = 20
    y[0] = -250
    while y[i] < 0:
        q[0] = dzdt(l[i], y[i], z[i])
        k[0] = dydt(z[i])
        q[1] = dzdt(l[i] + h / 2, y[i] + k[0] * h / 2, z[i] + q[0] * h / 2)
        k[1] = dydt(z[i] + q[0] * h / 2)
        q[2] = dzdt(l[i] + h / 2, y[i] + k[1] * h / 2, z[i] + q[1] * h / 2)
        k[2] = dydt(z[i] + q[1] * h / 2)
        q[3] = dzdt(l[i] + h, y[i] + k[2] * h, z[i] + q[2] * h)
        k[3] = dydt(z[i] + q[2] * h)
        z[i + 1] = z[i] + h * (q[0] + 2 * q[1] + 2 * q[2] + q[3]) / 6
        y[i + 1] = y[i] + h * (k[0] + 2 * k[1] + 2 * k[2] + k[3]) / 6
        l[i + 1] = l[i] + h
        textBox1.Text += f"Шаг = {i}\r"
        textBox1.Text += f"l[{i + 1}] = {l[i + 1]}\r"
        textBox1.Text += f"q[0] = {q[0]}\r"
        textBox1.Text += f"k[0] = {k[0]}\r"
        textBox1.Text += f"q[1] = {q[1]}\r"
        textBox1.Text += f"k[1] = {k[1]}\r"
        textBox1.Text += f"q[2] = {q[2]}\r"
        textBox1.Text += f"k[2] = {k[2]}\r"
        textBox1.Text += f"q[3] = {q[3]}\r"
        textBox1.Text += f"k[3] = {k[3]}\r"
        textBox1.Text += f"z[{i + 1}] = {z[i + 1]}\r"
        textBox1.Text += f"y[{i + 1}] = {y[i + 1]}\r\r"
        i += 1
    j = i
    count = 0
    for i in range(j + 1):
        if i % 2 == 0:
            sum2 += y[i]
```

```
            count += 1
        else:
            sum += y[i]
    sum /= (j - count + 1)
    sum2 /= count
    textBox5.Text += f"Сумма y[i],h = {sum2}\r"
    textBox5.Text += f"Сумма y[i],h/2 = {sum}\r"
    textBox5.Text += f"Погрешность по правилу Рунге-Кутты = |y[i],h - y[i],h/2|/15
= {abs(sum - sum2) / 15}"
```

Блок-схема метода численного дифференцирования

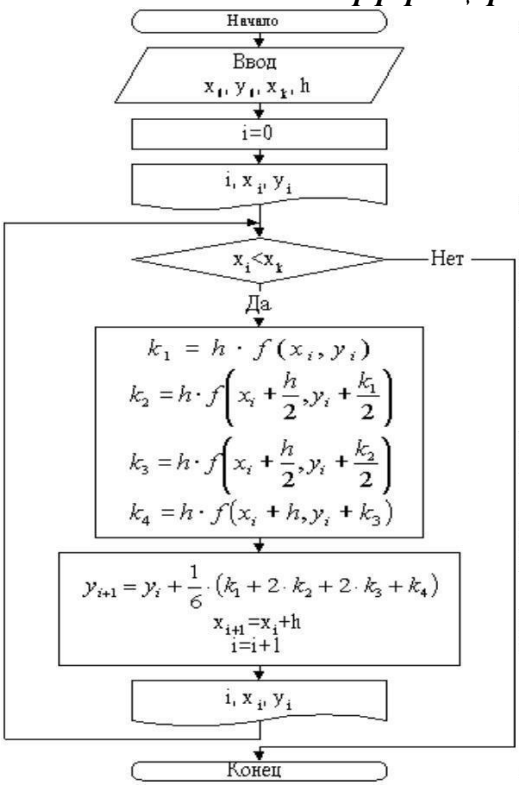
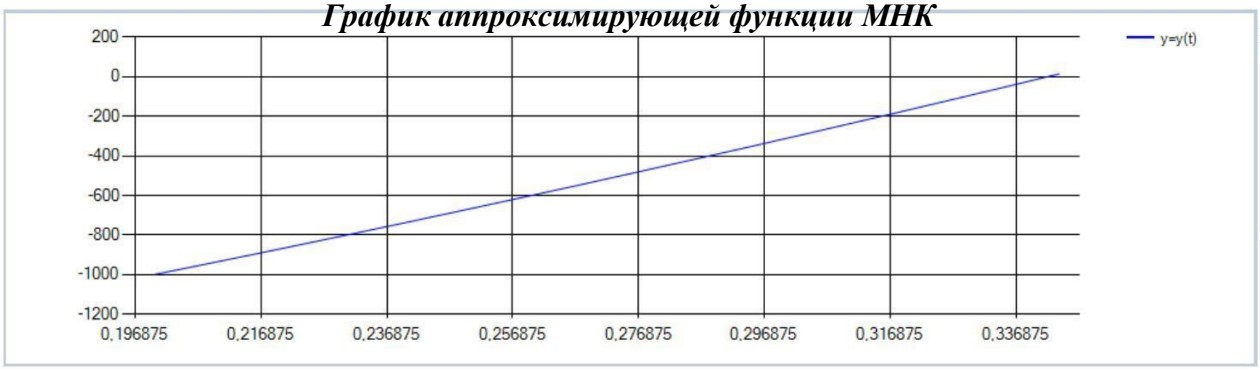


График аппроксимирующей функции МНК



Аппроксимирующая функция
y = 0,719*t^2 + 56,982*t + -2095,86

T = 24,7 сек, L = (1975,6;15,127)

Погрешности численного метода дифференцирования

Среднее квадратичное отклонение для квадратичной функции = 9,08345125428924E-15

Погрешность по правилу Рунге-Кутты = |y[i],h - y[i],h/2|/15 = 0,0469541687880906

Для квадратичной зависимости СКО: $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(y_i-(ax_i^2+bx_i+c))^2}{n-1}}$=9,08345125428924E-15

Погрешность по правилу Рунге-Кутта: $\frac{\max|y_{i,h}^{[\,]} - y_{i,\frac{h}{2}}^{[\,]}|}{15}$= 0,0469541687880906

1. ***Численное дифференцирование:***

Данную операцию приходится выполнять в тех случаях, когда информация об исследуемой функции задана в табличной форме или в виде набора результатов экспериментальных исследований. К численному дифференцированию прибегают также и в случае, когда задан явный аналитический вид функции, но выражение для производной оказывается достаточно сложным и желательно его заменить более простым.

Задача численного дифференцирования состоит в приближенном вычислении значения производной функции $y = f(x)$ в некоторой точке x^* по заданным в конечном числе точек (узлах сетки) значениям этой функции $y_i = f(x_i)$.

2. ***Явный и неявный метод Эйлера:***

Метод Эйлера — простейший численный метод решения систем ОДУ. Он является явным, одношаговым методом первого порядка точности, основан на аппроксимации интегральной кривой кусочно-линейной функцией, так называемой ломаной Эйлера. Состоит в приближенной замене производной $y' = f(x, y)$ в уравнении ее разностным аналогом. Тогда дифференциальное уравнение можно записать в виде:

$$\frac{y_{i+1} - y_i}{x_{i+1} - x_i} = f(x_i, y_i)$$

Расчетная формула метода Эйлера имеет вид:

$$y_{i+1} = y_i + hf(x_i, y_i)$$

$$x_{i+1} = x_i + h\ (i = 0,1,2, \dots, n-1)$$

Неявный метод Эйлера — это способ интегрирования, хорошо подходящий для интегрирования жёстких уравнений, которые при других методах становятся нестабильными. Его недостаток заключается в том, что он требует решения системы уравнений на каждом шаге времени. При построении неявного метода Эйлера значение функции $f(x, y)$ вычисляется на $i + 1$ шаге, т.е. для решения задачи Коши используется следующее выражение:

$$y_{i+1} = y_i + hf(x_{i+1}, y_{i+1})$$

$$x_{i+1} = x_i + h\ (i = 0,1,2, \dots, n-1)$$

Таким образом, для нахождения приближенного значения искомой функции на $i + 1$ шаге необходимо решить нелинейное уравнение относительно y_{i+1}, для решения которого необходимо использовать численные методы, например, метод Ньютона.:

$$y_{i+1} - hf(x_{i+1}, y_{i+1}) - y_i$$

3. ***Погрешности численного дифференцирования:***

Численное решение сильно отличается от точного и главный вопрос при использовании метода Эйлера или любого другого численного метода состоит в оценке точности приближенных значений. Вообще говоря, существуют два источника погрешности этих приближений:

1) ***ошибка дискретизации***, возникающая в результате замены дифференциального уравнения разностной аппроксимацией;

2) ***ошибка округления***, накопившаяся при выполнении арифметических операций по формулам.

$E(h) = \max|y_k - y(x_k)|$ - *глобальная ошибка дискретизации,* где $E(h)$ зависит от h.

$L(h) = O(h^n)$ - *порядок локальной ошибки дискретизации,* где зависит от величины шага h, вида правой части дифференциального уравнения и от выбора отрезка $[x_0, x]$.

4. ***Метод Рунге-Кутта 4-го порядка точности:***

Метод Рунге-Кутта 4-го порядка является одношаговым с явной схемой, позволяют получить большую точность, достаточно распространен.

Пусть на отрезке $[a, b]$ требуется найти приближенное численное решение дифференциального уравнения $y' = f(x, y)$ с начальными условиями $y(x_0) = y_0$. Разбиваем отрезок $[a, b]$ на n равных частей точками $x_i = x_0 + ih\ (i = 0,1,2, \dots, n), h = \frac{b-a}{h}$ - шаг интегрирования (решения) дифференциального уравнения.

Каждое последующее приближенное значение yi+1 искомого решения y определяется через текущее значение y_i с помощью рекуррентных формул по вычислительной схеме:

$$y_{i+1} = y_i + \Delta y_i$$

$$\Delta y_i = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

$$k_1 = hf(x_i, y_i)$$

$$k_2 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_1}{2}\right)$$

$$k_3 = hf\left(x_i + \frac{h}{2}, y_i + \frac{k_2}{2}\right)$$

$$k_4 = hf(x_i + h, y_i + k_3)$$

5. ***Метод «прогноз-коррекция»:***

Методами «прогноза и коррекции» (предсказывающе-исправляющими / схемами «предиктор-корректор») в вычислительной математике называют группу методов численного решения различных задач, у которых каждый шаг состоит из двух основных действий: первое (предиктор) заключается в вычислении грубого начального приближения искомой величины, выполняется однократно, второе действие (корректор) уточняет его.

Модифицированный метод Эйлера:

$$y_{i+1}^{(0)} = y_i + hf(x_i, y_i) - \textit{прогноз,}\ \text{где}\ \frac{dy}{dx} = f(x_i, y_i)$$

$$y_{i+1}^{(k+1)} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_{i+1}, y_{i+1})) - \textit{коррекция}$$