

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский государственный технологический университет «СТАНКИН» (ФГБОУ ВО «МГТУ «СТАНКИН»)**

**Институт Кафедра прикладной**

**информационных систем математики и технологий**

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

[Краткая](#_bookmark0) [теория](#_bookmark0) [3](#_bookmark0)

[Практическое](#_bookmark1) [задание](#_bookmark1) [4](#_bookmark1)

Система ДУ 4

[Численное](#_bookmark1) [решение](#_bookmark1) [системы](#_bookmark1) [методом](#_bookmark1) [Рунге-Кутта](#_bookmark1) [4](#_bookmark1)

Код программы на Python 6

[Блок-схема](#_bookmark2) [метода](#_bookmark2) [численного](#_bookmark2) [дифференцирования](#_bookmark2) [10](#_bookmark2)

[График](#_bookmark2) [аппроксимирующей](#_bookmark2) [функции](#_bookmark2) [МНК](#_bookmark2) [10](#_bookmark2)

[Погрешности](#_bookmark2) [численного](#_bookmark2) [метода](#_bookmark2) [дифференцирования](#_bookmark2) [10](#_bookmark2)

[Анализ полученных результатов 11](#_TOC_250000)

***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).

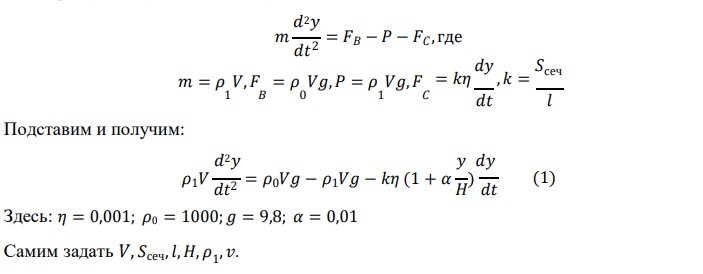
# Краткая теория

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



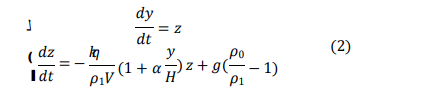
По оси абсцисс

По второму закону Ньютона получим:

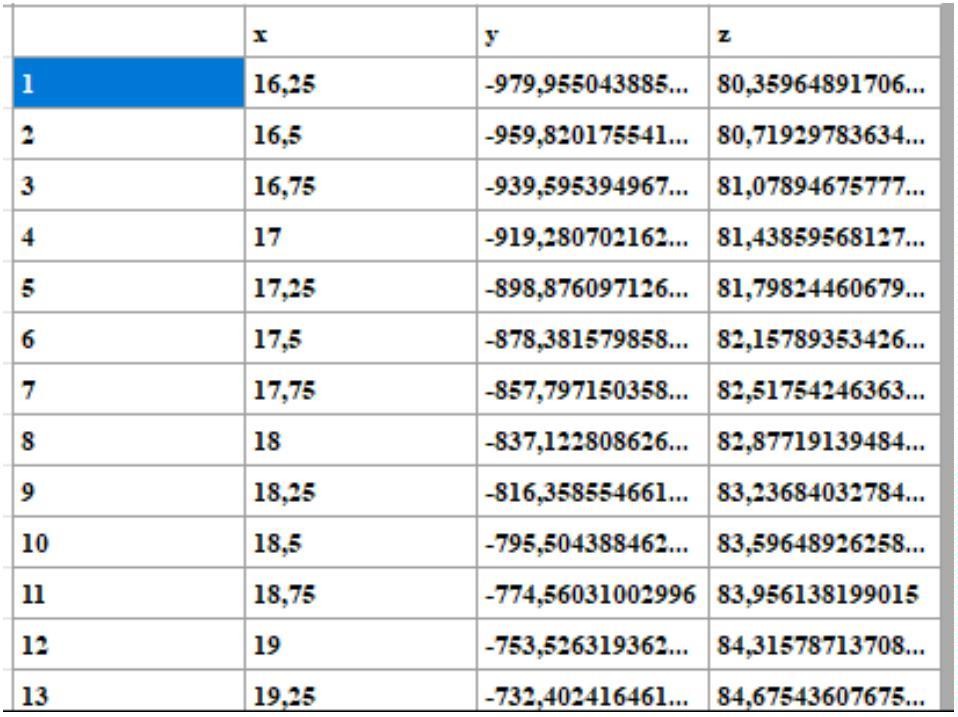


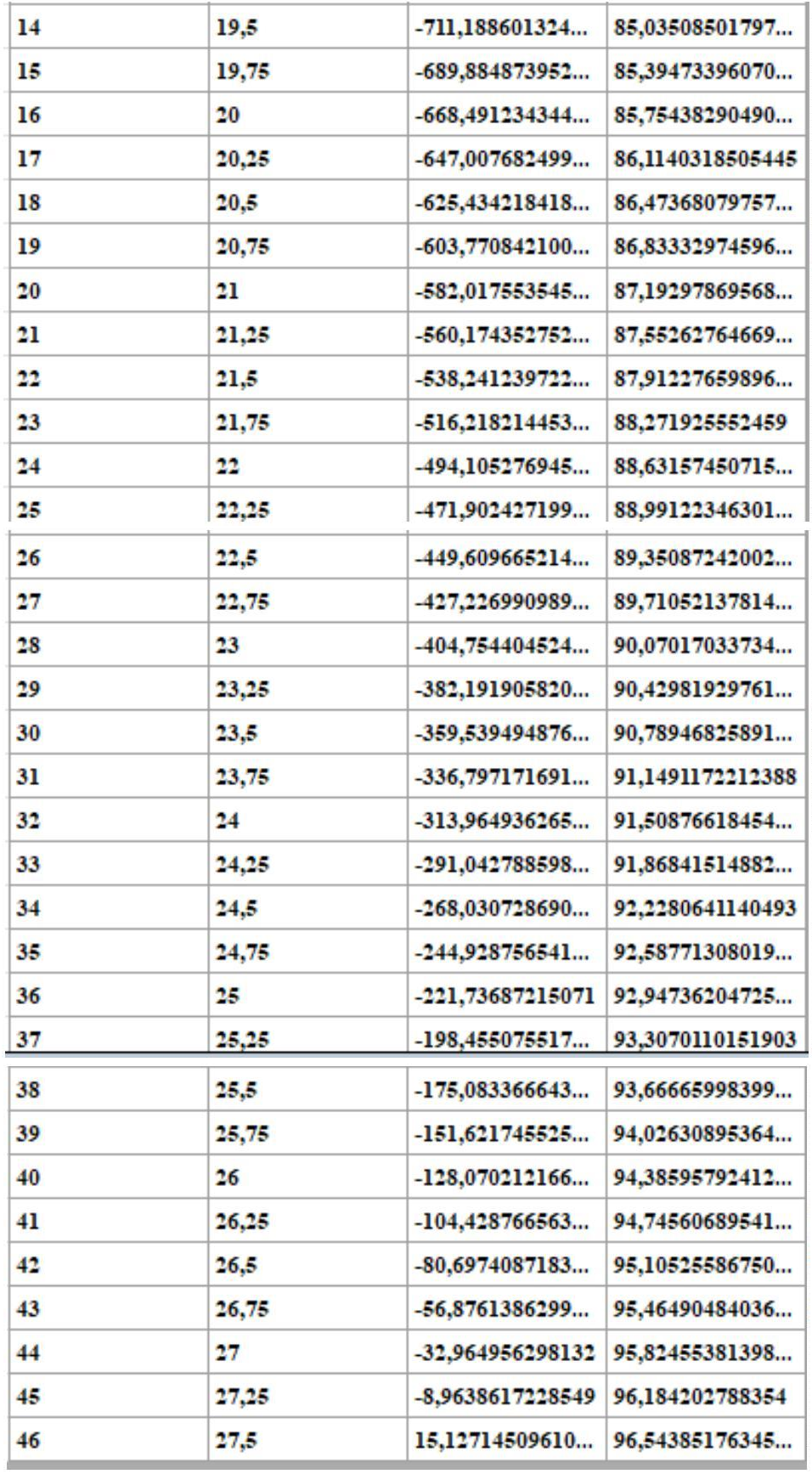
# Практическое задание

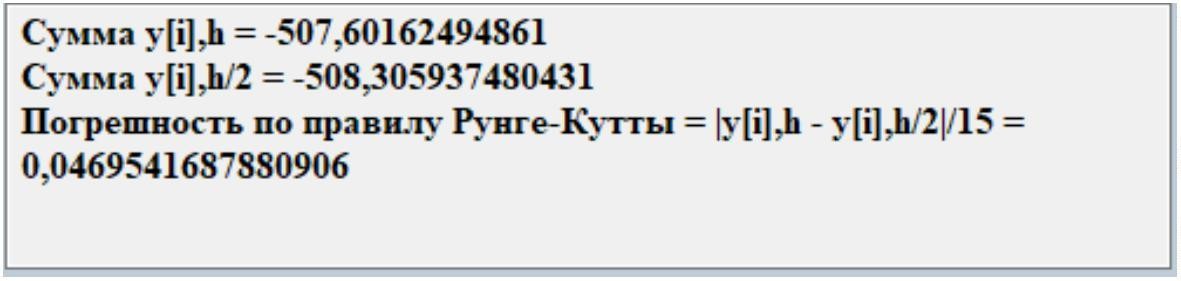
***Система дифференциальных уравнений***



### Численное решение системы методом Рунге-Кутта

******



******

***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 Н(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

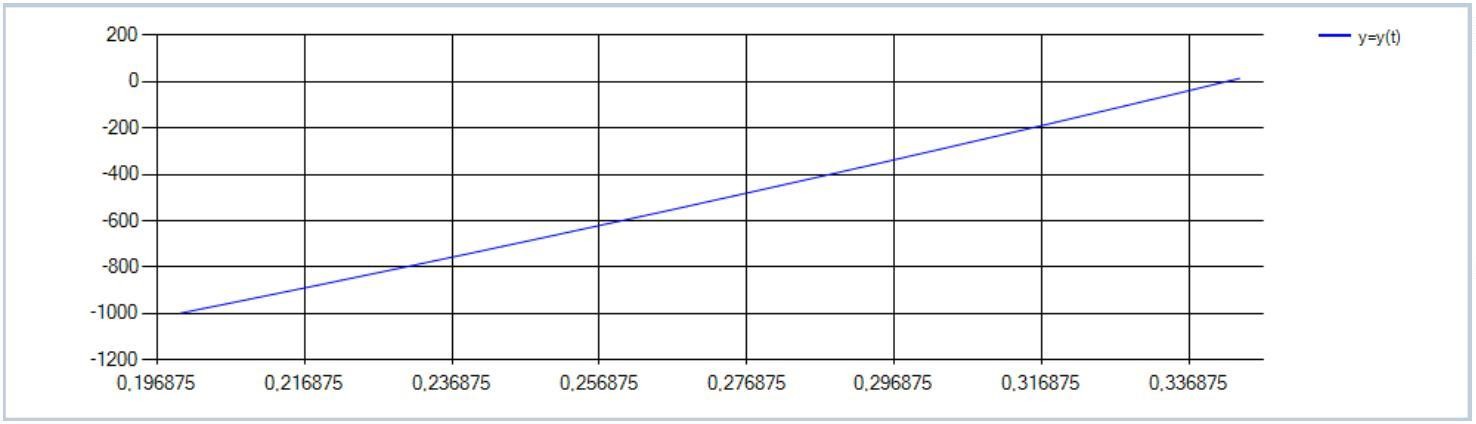
textBox5.Text += f"Сумма y[i],h = {sum2}\r" textBox5.Text += f"Сумма y[i],h/2 = {sum}\r"

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

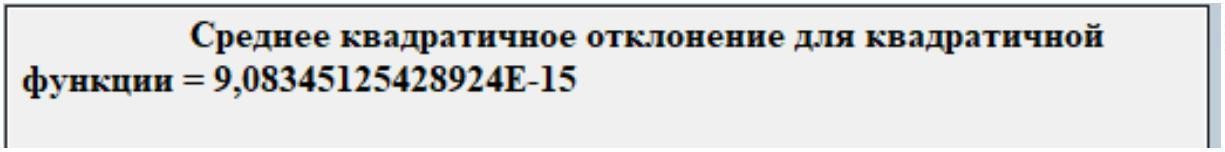
******



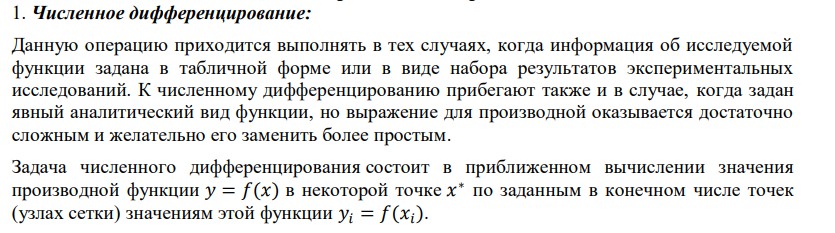
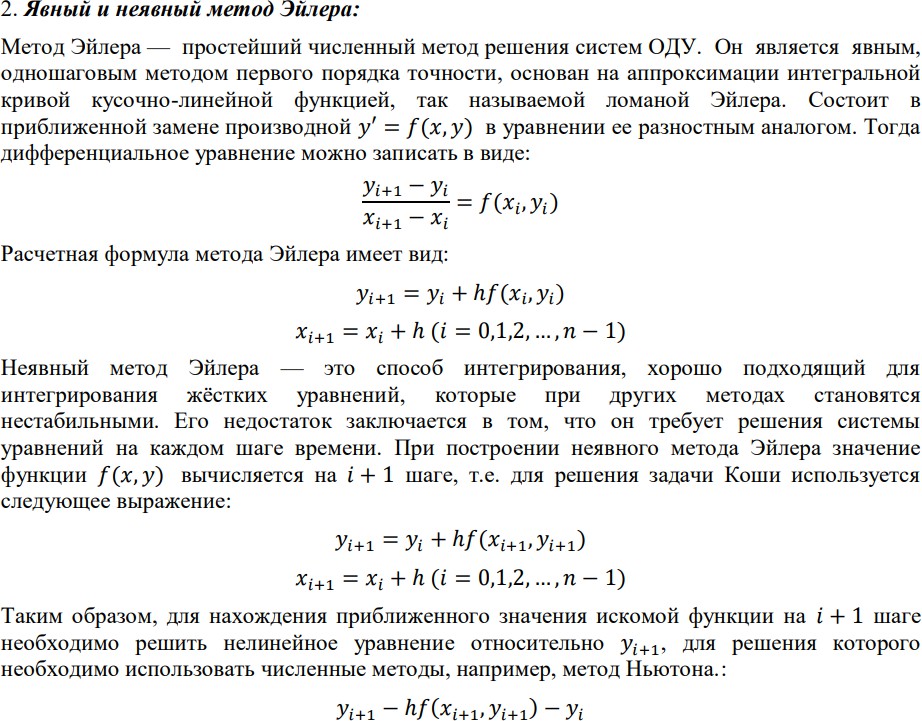
***График аппроксимирующей функции МНК***

T = 24,7 сек, L = (1975,6;15,127)

### Погрешности численного метода дифференцирования

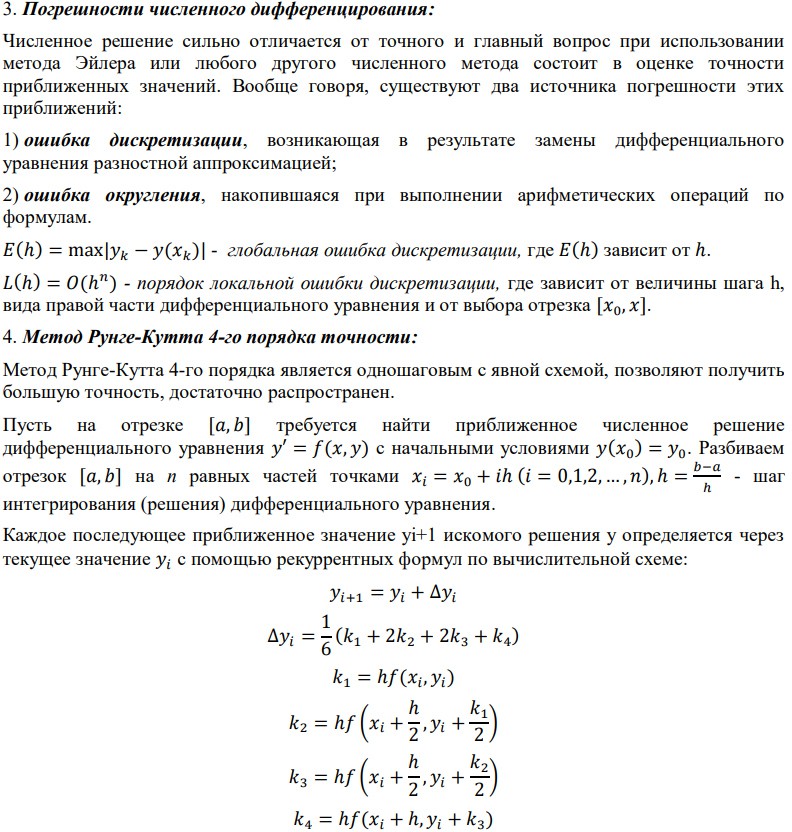
******

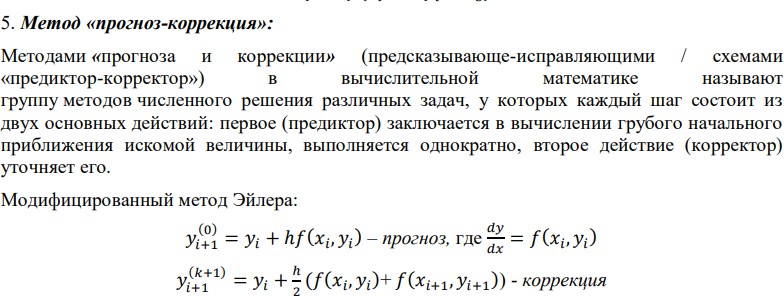
## Анализ полученных результатов



***Вывод:*** была решена система ДУ методом Рунге-Кутта 4-го порядка точности, полученные значения были аппроксимированы по МНК, благодаря чему была найдена траектория всплытия подводной – парабола.

**Контрольные вопросы**



****