

# Цель работы

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

# Постановка задачи.

H- глубина известна, Т-время всплытия, L- точка всплытия,

они неизвестны, их нужно найти.

По оси абсцисс =v, x=vt

Тогда: L=vT

По второму закону Ньютона получим:

Подставим и получим:

Здесь: η =0,001, ρ0 =1000 ,g=9,8 ,α=0,01

Самим задать

H0 = -350 м, l = 73,8, Sсеч = 718,73 м2, Vо = 2668,3 м3, m = 3040000 кг, P1 =863, Vс = 7,5 , k = 10 м

Получим систему:

Решая её в программе методом Рунге- Кутта получаем следующие точки y траектории в зависимости от времени.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0 | 7.5 | 15 | 22.5 | 30 | 37.5 | 45 | 52.5 | 60 | 67.5 | 75 |
| y | -350 | -349.222 | -346.889 | -342.999 | -337.554 | -330.553 | -321.997 | -311.885 | -300.216 | -286.993 | -272.213 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 82.5 | 90 | 97.5 | 105 | 112.5 | 120 | 127.5 | 135 | 142.5 | 150 |
| -255.878 | -237.987 | -218.54 | -197.538 | -174.98 | -150.866 | -125.196 | -97.9709 | -69.1898 | -38.8529 |

|  |
| --- |
| 157.5 |
| -6.96036 |

По методу наименьших квадратов находим уравнение, аппроксимирующее данные точки:

Y= 0.0138287x2+ 0.00000133514x-350

Код программы:

#include <iostream>

using namespace std;

float

H = -350, l = 73.8, S = 718.73, V = 2668.3, m = 3040000,

p1 = 863, v = 7.5, h = 0.001, p0 = 1000,

g = 9.8, a = 0.01, k = 10;

float\*\* izm = new float\* [2];

float dydt(float z) {

return z;

}

float dzdt(float y, float z) {

return (float)(((-(h \* k) / (V \* p1)) \* (1 + a \* (y / H))) \* z + g \* ((p0 / p1) - 1));

}

float n = 1;

int c = 0;

void rungeKutta(float y0, float z0)

{

float k0, k1, k2, k3, q0, q1, q2, q3;

float y1, z1;

q0 = dzdt(y0, z0); k0 = dydt(z0);

q1 = dzdt((float)(y0 + k0 \* (n / 2)), (float)(z0 + q0 \* (n / 2)));

k1 = dydt((float)(z0 + q0 \* (n / 2)));

q2 = dzdt((float)(y0 + k1 \* (n / 2)), (float)(z0 + q1 \* (n / 2)));

k2 = dydt((float)(z0 + q1 \* (n / 2)));

q3 = dzdt((float)(y0 + k2 \* n), (float)(z0 + q2 \* n));

k3 = dydt((float)(z0 + q2 \* n));

z1 = z0 + (float)(n / 6) \* (q0 + 2 \* q1 + 2 \* q2 + q3);

y1 = y0 + (float)(n / 6) \* (k0 + 2 \* k1 + 2 \* k2 + k3);

if (y1 < 0) {

c++;

izm[0][c] = c \* v;

izm[1][c] = y1;

rungeKutta(y1, z1);

}

}

void MNK() {

float\*\* A2 = new float\* [3];

for (int i = 0; i < 3; i++)

A2[i] = new float[4];

for (int i = 0; i < 3; i++) {

for (int j = 0; j < 4; j++) {

A2[i][j] = 0;

}

}

for (int i = 0; i <= c; i++) {

A2[0][0] += pow(izm[0][i], 4);

A2[0][1] += pow(izm[0][i], 3);

A2[0][2] += pow(izm[0][i], 2);

A2[1][0] += pow(izm[0][i], 3);

A2[1][1] += pow(izm[0][i], 2);

A2[1][2] += izm[0][i];

A2[2][0] += pow(izm[0][i], 2);

A2[2][1] += izm[0][i];

A2[2][2] = c + 1;

A2[0][3] += izm[0][i] \* izm[0][i] \* izm[1][i];

A2[1][3] += izm[0][i] \* izm[1][i];

A2[2][3] += izm[1][i];

}

float\* X = new float[3];

float d;

for (int i = 0; i < 3; i++) // прямой ход

{

for (int j = i + 1; j < 3; j++)

{

d = A2[j][i] / A2[i][i];

for (int k = i; k < 3; k++)

{

A2[j][k] = A2[j][k] - d \* A2[i][k];

}

A2[j][3] = A2[j][3] - d \* A2[i][3];

}

}

for (int j = 0; j < 3; j++)

for (int i = 3; i >= j; i--)

A2[j][i] = A2[j][i] / A2[j][j];

for (int i = 3 - 1; i >= 0; i--) // обратный ход

{

float s;

d = 0;

for (int j = i + 1; j < 3; j++)

{

s = A2[i][j] \* X[j];

d = d + s;

}

if (A2[i][i] != 0)

X[i] = (A2[i][3] - d) / A2[i][i];

else {

X = NULL;

}

}

cout << "\nкоэфициенты аппроксимирующего уравнения: ";

cout << "\na " << X[0] << endl;

cout << "b " << X[1] << endl;

cout << "c " << X[2] << endl;

}

int main()

{

setlocale(LC\_ALL, "ru");

for (int i = 0; i < 2; i++)

izm[i] = new float[28];

izm[0][0] = 0;

izm[1][0] = -350;

float y0 = -350, z0 = 0;

rungeKutta(y0, z0);

cout << "координаты точек:\n ";

for (int i = 0; i <= c; i++) {

cout << "(" << izm[0][i] << ", ";

cout << izm[1][i] << "), ";

}

MNK();

return 0;

}

Найденные точки и график аппроксимирующей функции рис.1

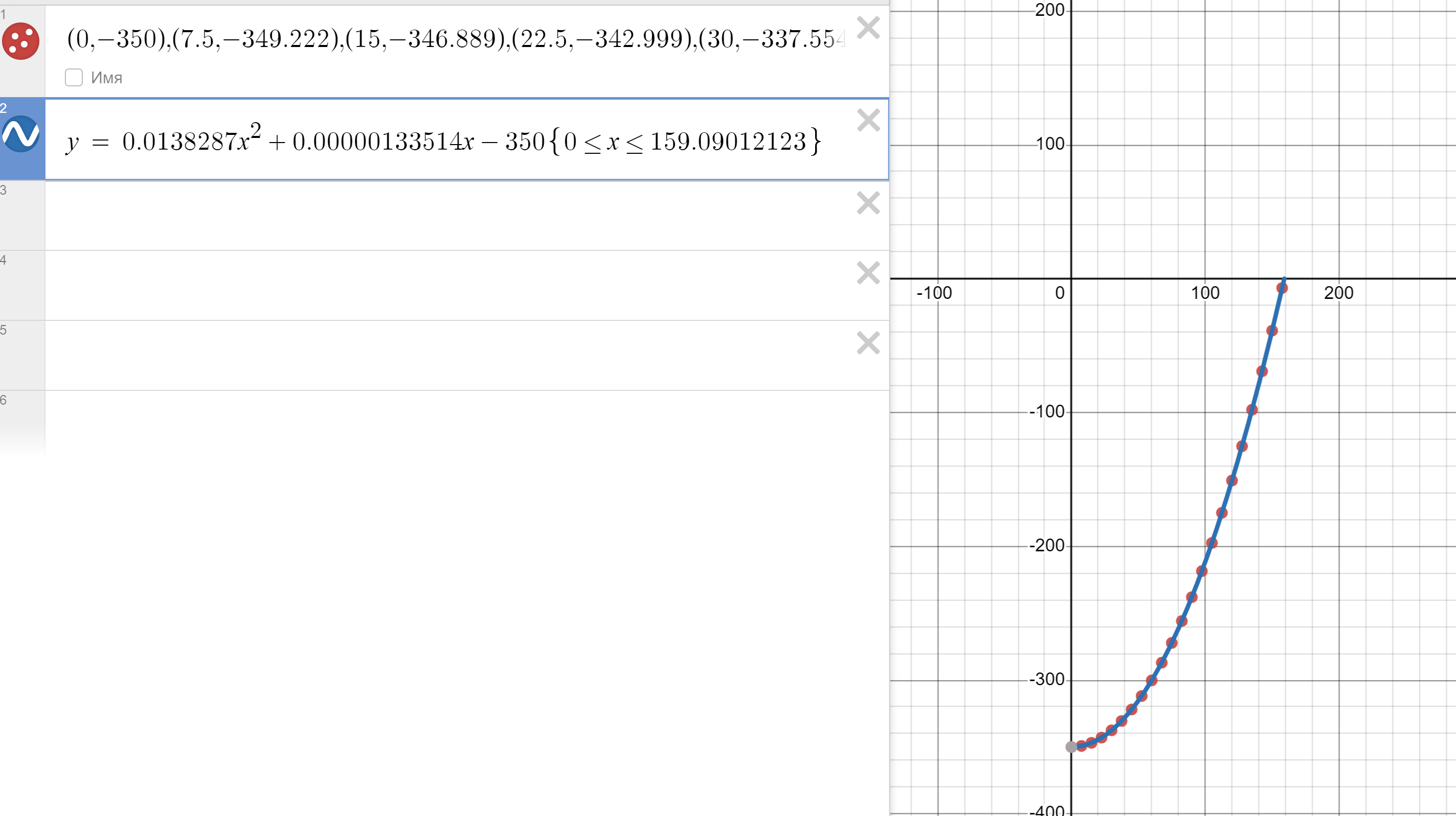


Рис. 1. Найденные точки и график аппроксимирующей функции

Погрешность аппроксимации: 0.0000000054

Погрешность метода Рунге-Кутта: 0.000000763

Найдём время T и расстояние L:

H=aT2+bT+c.

H=0.0138287T2-0.00000133514T-350

T=21.21203c

L=159.09022м

# Вывод

На данной лабораторно й работе мы изучили методы численного дифференцирования для решения системы обыкновенных дифференциальных уравнений и применили их на практике для определения траектории всплытия подводной лодки.