**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

ЗВІТ

про виконання лабораторної роботи №1

з дисципліни «Комп’ютерне моделювання складних процесів і систем»

Виконав:

студент групи ПА-18-1

Щербак Роман

Викладач:

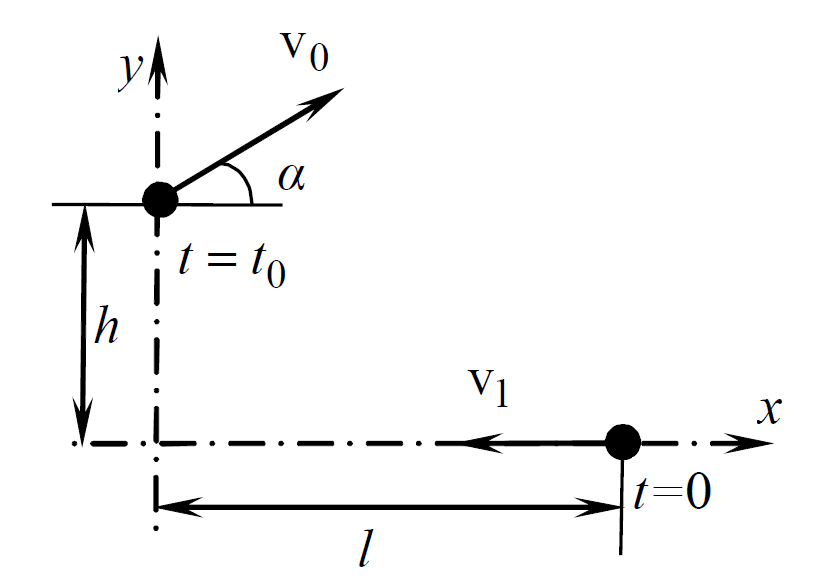
Ю. Я. Годес

Дніпро

2022

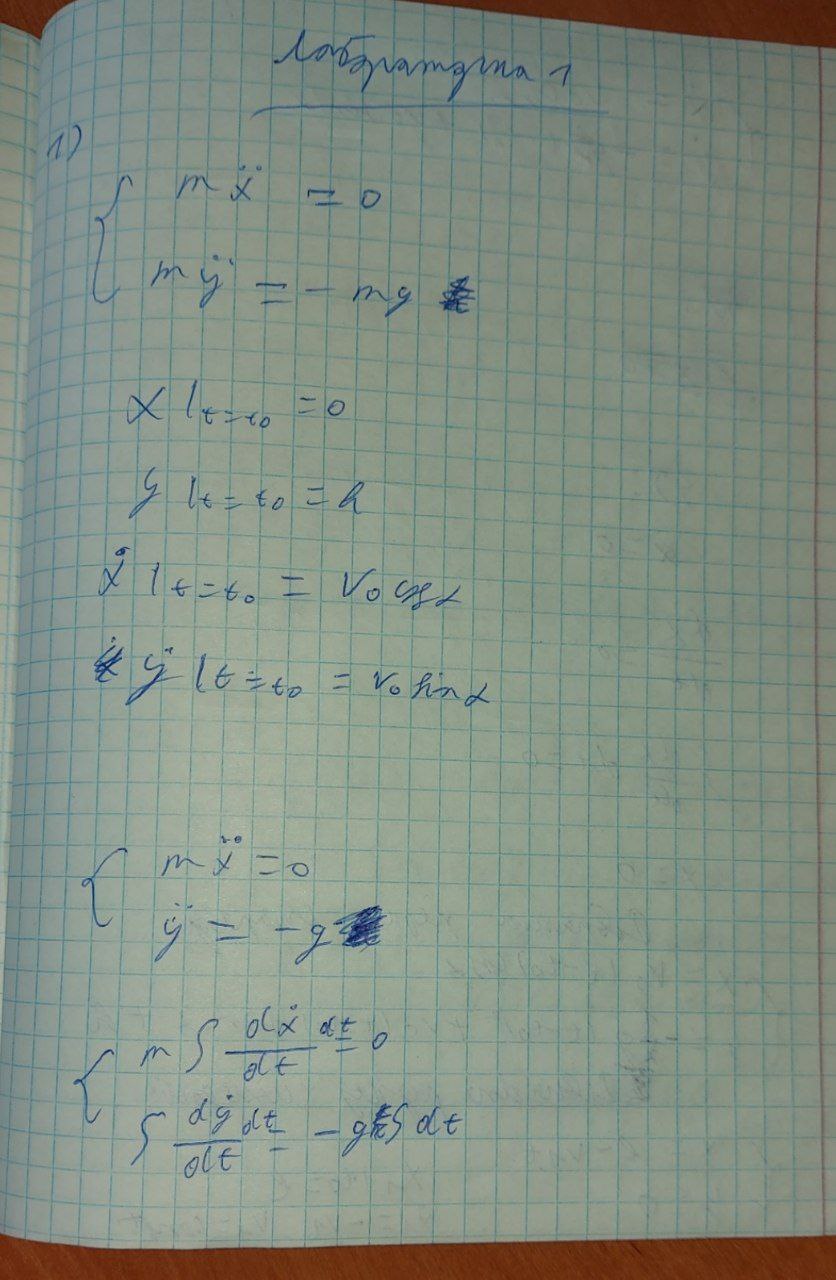
1. **Постановка завдання (Варіант 1)**

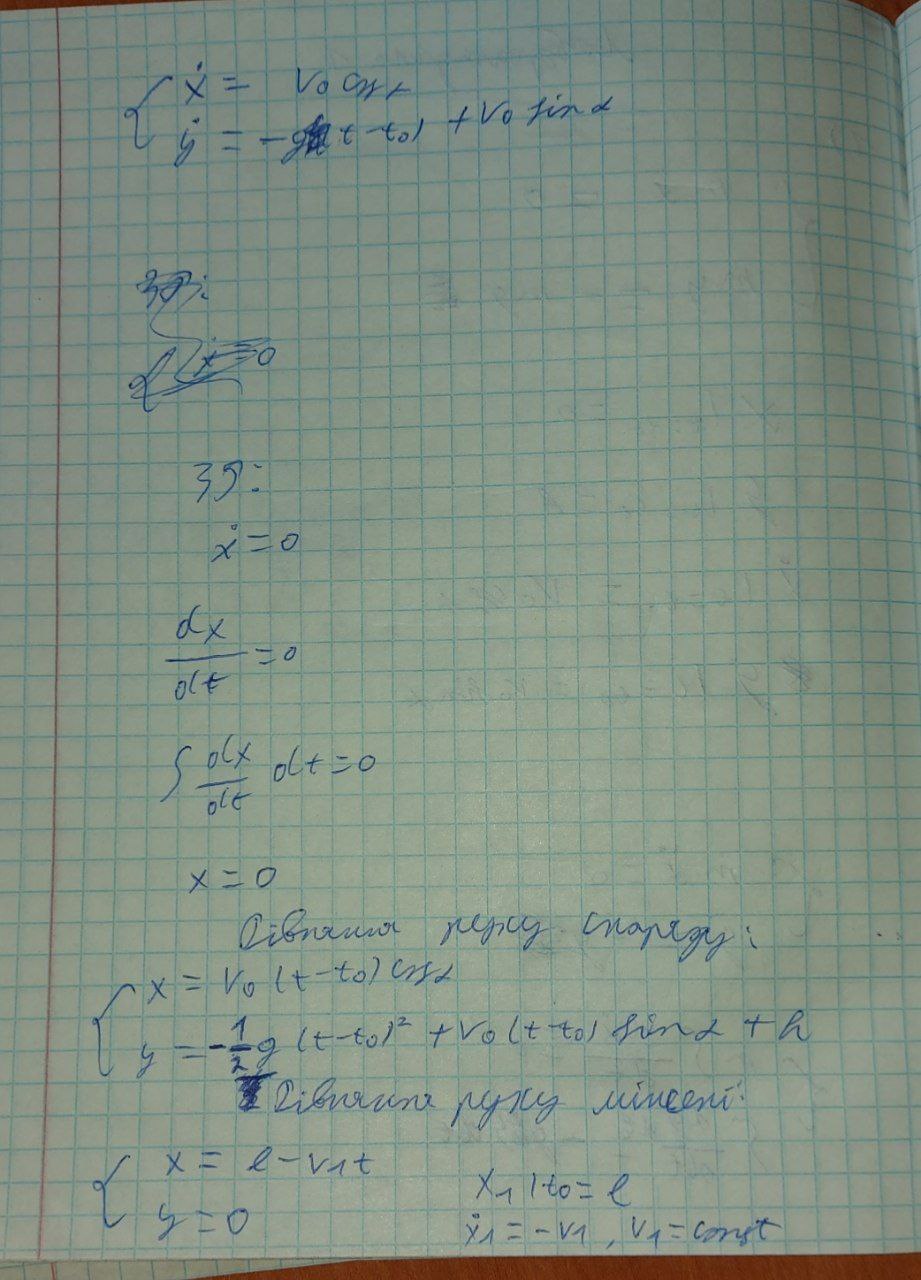
* Траєкторія мішені, що рухається зі сталою швидкістю , є прямою, паралельною до осі абсцис. У початковий момент часу t=0 мішень знаходиться на відстані від осі ординат, що спрямована вертикально вгору. У деякий момент часу зі зброї відбувається постріл. Початкова швидкість снаряду дорівнює та утворює кут з віссю абсцис.



* Скласти рівняння руху снаряду та мішені без опору повітря.
* Розробити комп’ютерну модель руху снаряду та мішені, що видає координати снаряду та мішені з певним кроком за часом. Зобразити рух графічно.
* Установити умову влучання снаряду в мішень. Дослідити можливість влучання, залежно від параметрів моделі.
* Побудувати ускладнену модель, зважаючи на опір повітря руху снаряду. Силу опору прийняти пропорційною до швидкості снаряду, коефіцієнт пропорційности дорівнює , де – маса снаряду. Величину віднести до вихідних даних моделі.

1. **Рівняння без опору повітря**



****

1. **Програма мовою Python**

* Модуль main.py

import moving\_equations as me

import matplotlib.pyplot as plt

import matplotlib.animation as plta

import plots as p

import parameters\_checking as pc

import math as m

import interface as inter

print()

def animate(i):

x\_1.append(shell\_moving\_x[i])

y\_1.append(shell\_moving\_y[i])

x\_2.append(target\_moving\_x[i])

y\_2.append(target\_moving\_y[i])

plt.plot(x\_1, y\_1, color = "orange")

plt.plot(x\_2, y\_2, color = "blue")

variables = inter.make\_interface()

try:

v\_0, v\_1, l, h, alpha = variables[0], variables[1], variables[2], variables[3], m.radians(variables[4])

print("Параметри системи:")

print(f"v\_0 = {v\_0}")

print(f"v\_1 = {v\_1}")

print(f"l = {l}")

print(f"h = {h}")

print(f"alpha = {m.degrees(alpha)}")

print()

t\_start = 0

delta\_t = 0.001

g = 9.81

t\_end, t\_0 = pc.get\_t(v\_0, v\_1, alpha, h, l, g)

print(f"Момент пострілу: {t\_0}")

print(f"Момент влучення: {t\_end}")

print()

if t\_0 >= 0 and t\_end >= 0:

h,l = pc.check\_height\_length(h, l)

t\_end, t\_0 = pc.get\_t(v\_0, v\_1, alpha, h, l, g)

hit, hit\_moment, shell\_moving\_x, shell\_moving\_y, target\_moving\_x, target\_moving\_y = me.calculate\_hit(t\_start, t\_end, delta\_t, l, t\_0, v\_0, v\_1, alpha, h, g)

print("Координати влучення:")

print(f"Снаряд: ({shell\_moving\_x[-1]};{shell\_moving\_y[-1]})")

print(f"Мішень: ({target\_moving\_x[-1]};{target\_moving\_y[-1]})")

x\_1, y\_1, x\_2, y\_2 = [], [], [], []

anim = plta.FuncAnimation(plt.gcf(), animate)

p.scatter\_points(shell\_moving\_x, shell\_moving\_y, target\_moving\_x, target\_moving\_y)

plt.show()

else:

print()

print("Умови додатності часу не виконано")

except:

print("Виникла неочікувана помилка, можливо введено неправильні дані.")

* Модуль moving\_equations.py

import math as m

def shell\_moving(t, shoot\_moment, velocity, angle, height, g):

if t-shoot\_moment < 0:

x = 0

y = height

else:

x = round(velocity\*m.cos(angle)\*(t-shoot\_moment), 6)

y = round((-g/2)\*((t-shoot\_moment)\*\*2) + velocity\*m.sin(angle)\*(t-shoot\_moment) + height, 6)

return x, y

def target\_moving(length, velocity, t):

x = round(length-velocity\*t, 6)

y = 0

return x, y

def calculate\_hit(t\_start, t\_end, delta\_t, length, shoot\_moment, velocity\_shell, velocity\_target, shoot\_angle, height, g):

t\_1 = t\_start

hit = False

shell\_moving\_x, shell\_moving\_y, target\_moving\_x, target\_moving\_y = [], [], [], []

x\_shell\_t, y\_shell\_t = 0, height

x\_target\_t, y\_target\_t = 0, length

i = 0

end = False

while t\_1 <= t\_end:

x\_shell\_t, y\_shell\_t = shell\_moving(t\_1, shoot\_moment, velocity\_shell, shoot\_angle, height, g)

x\_target\_t, y\_target\_t = target\_moving(length, velocity\_target, t\_1)

if i%100 == 0 or t\_1 == t\_end:

shell\_moving\_x.append(x\_shell\_t)

shell\_moving\_y.append(y\_shell\_t)

target\_moving\_x.append(x\_target\_t)

target\_moving\_y.append(y\_target\_t)

if t\_1 + delta\_t <= t\_end or end == True:

t\_1 += delta\_t

elif end == False:

t\_1 = t\_end

end = True

i += 1

return hit, t\_1, shell\_moving\_x, shell\_moving\_y, target\_moving\_x, target\_moving\_y

* Модуль parameters\_checking.py

import math as m

def check\_t\_0(t\_0, t\_start, t\_end):

if t\_0 < t\_start or t\_0 > t\_end:

t\_0 = (t\_0-t\_end) / 2

return t\_0

def check\_height\_length(height, length):

if height == 0:

height += 10

if length == 0:

length += 10

return height, length

def get\_t(velocity\_shell, velocity\_target, shoot\_angle, height, length, g):

l\_v = (length/velocity\_target)

sqrt\_parm = (velocity\_shell\*m.sin(shoot\_angle))+(((velocity\_shell\*\*2\*m.sin(shoot\_angle)\*\*2)+(2\*g\*height))\*\*(1/2))

divisor = velocity\_target\*g

t\_end = l\_v - ((velocity\_shell\*m.cos(shoot\_angle)\*sqrt\_parm)/divisor)

t\_0 = l\_v - (((velocity\_shell\*m.cos(shoot\_angle)+velocity\_target)\*sqrt\_parm)/divisor)

return t\_end, t\_0

* Модуль plots.py

import matplotlib.pyplot as plt

def scatter\_points(shell\_move\_x, shell\_move\_y, target\_move\_x, target\_move\_y):

plt.scatter(shell\_move\_x[0], shell\_move\_y[0], color = 'orange')

plt.scatter(target\_move\_x[0], target\_move\_y[0], color = 'blue')

plt.scatter(shell\_move\_x[-1], shell\_move\_y[-1], color = 'orange')

plt.scatter(target\_move\_x[-1], target\_move\_y[-1], color = 'blue')

* Модуль interface.py

import tkinter as tk

def button\_func(arr\_entr, arr\_var, root):

i = 0

while i < len(arr\_entr):

try:

arr\_var.append(float(arr\_entr[i].get()))

except ValueError:

root\_1 = tk.Tk()

root\_1.title("ПОМИЛКА")

root\_1.minsize(400, 100)

label = tk.Label(root\_1, text = "Не всі параметри є числом.")

label.pack()

i += 1

root.destroy()

def make\_interface():

root = tk.Tk()

root.title("Параметри системи")

root.minsize(600, 400)

entry\_arr = []

label\_1, entry\_1 = create\_label\_entry("v\_0: ", root)

entry\_1.insert(tk.END, "5")

pack\_label\_entry(entry\_arr, label\_1, entry\_1)

label\_2, entry\_2 = create\_label\_entry("v\_1: ", root)

entry\_2.insert(tk.END, "12")

pack\_label\_entry(entry\_arr, label\_2, entry\_2)

label\_3, entry\_3 = create\_label\_entry("l: ", root)

entry\_3.insert(tk.END, "30")

pack\_label\_entry(entry\_arr, label\_3, entry\_3)

label\_4, entry\_4 = create\_label\_entry("h: ", root)

entry\_4.insert(tk.END, "10")

pack\_label\_entry(entry\_arr, label\_4, entry\_4)

label\_5, entry\_5 = create\_label\_entry("alpha: ", root)

entry\_5.insert(tk.END, "45")

pack\_label\_entry(entry\_arr, label\_5, entry\_5)

variables = []

submit\_button = tk.Button(root, text='Задати параметри', command = lambda: button\_func(entry\_arr, variables, root))

submit\_button.pack()

root.mainloop()

return variables

def create\_label\_entry(text, root):

label = tk.Label(root, text=text)

entry = tk.Entry(root)

return label, entry

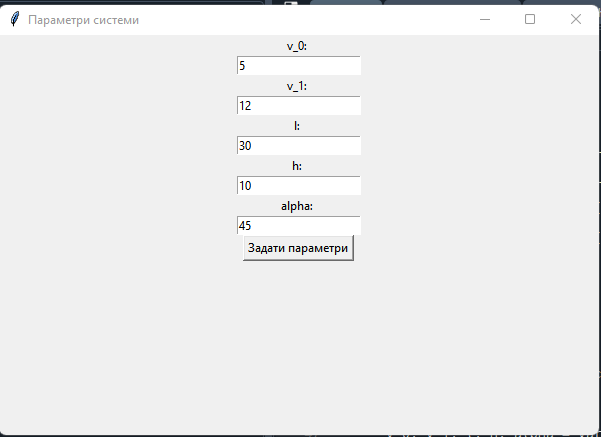
def pack\_label\_entry(entries\_arr, label, entry):

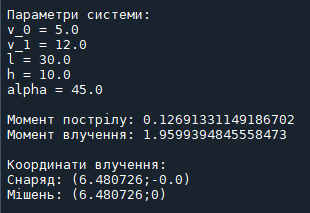
label.pack()

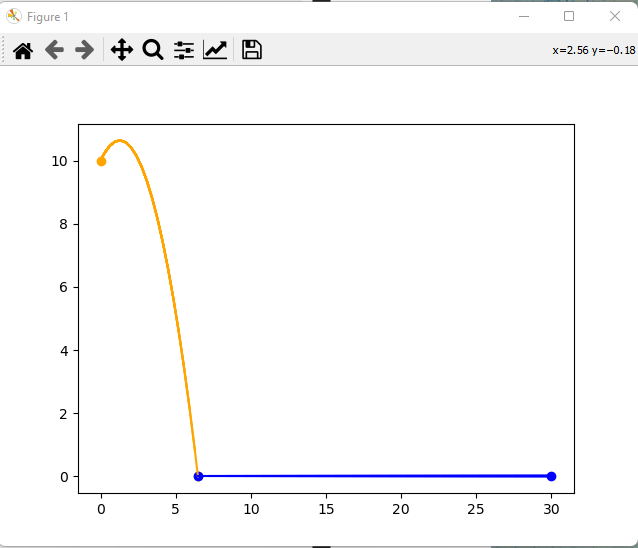
entry.pack()

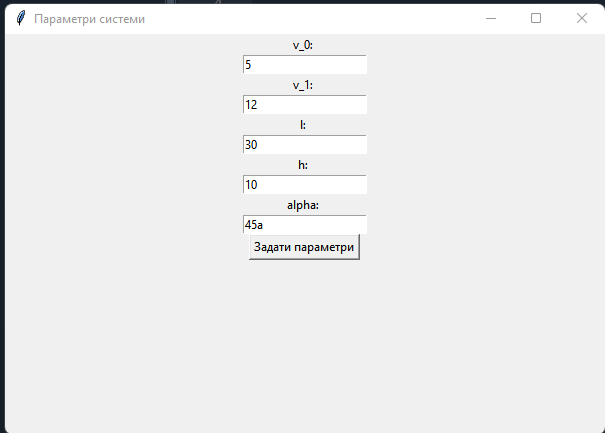
entries\_arr.append(entry)

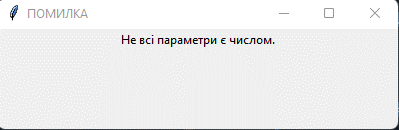
* Скриншоти роботи програми

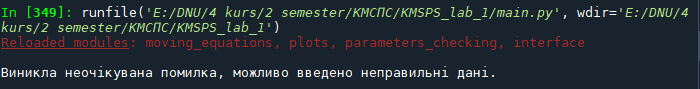


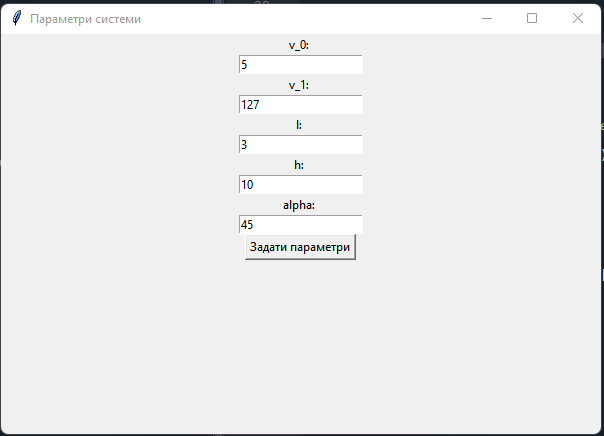


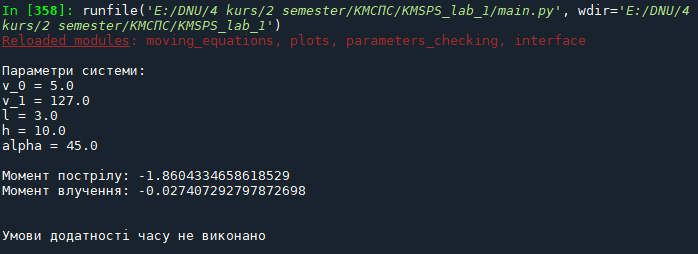




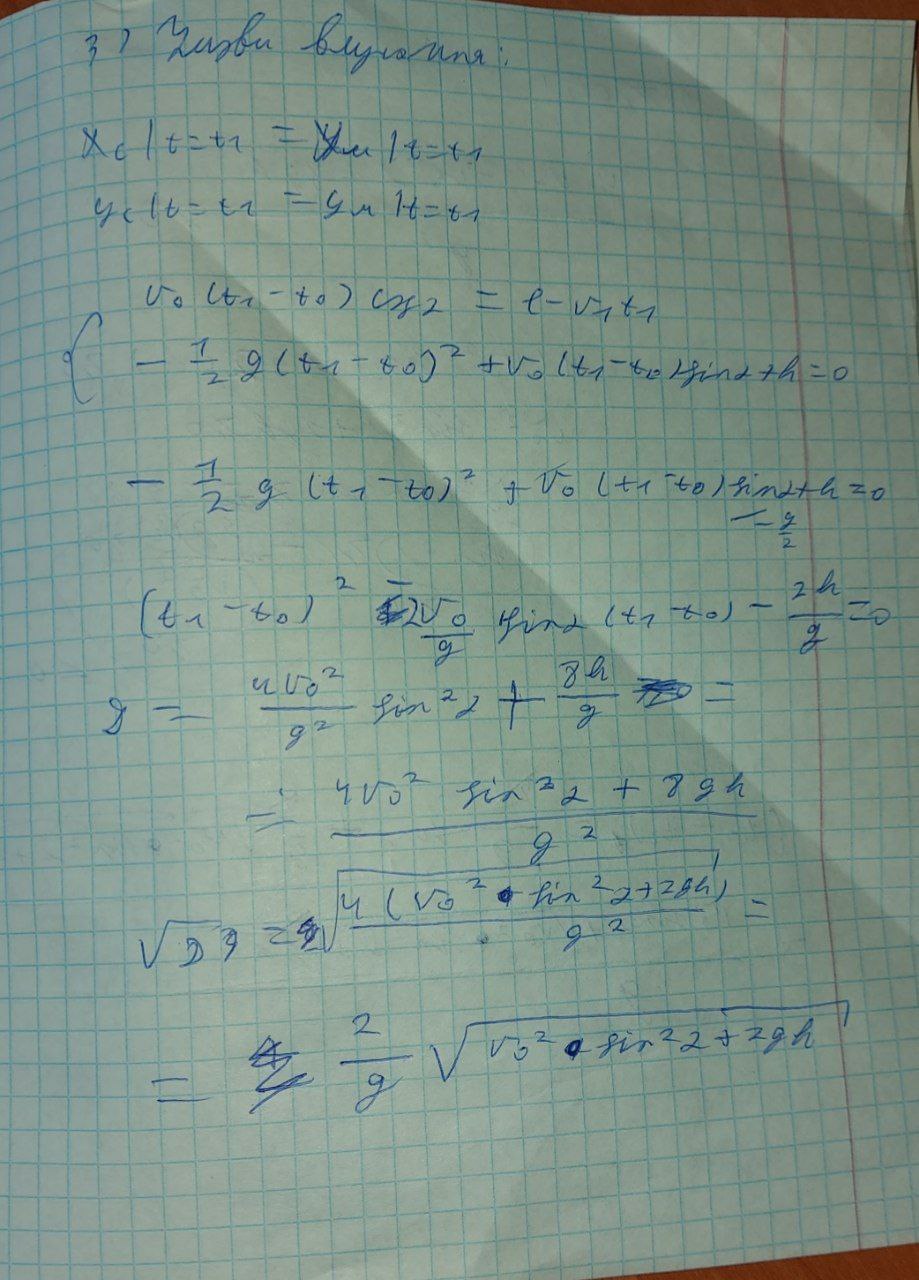


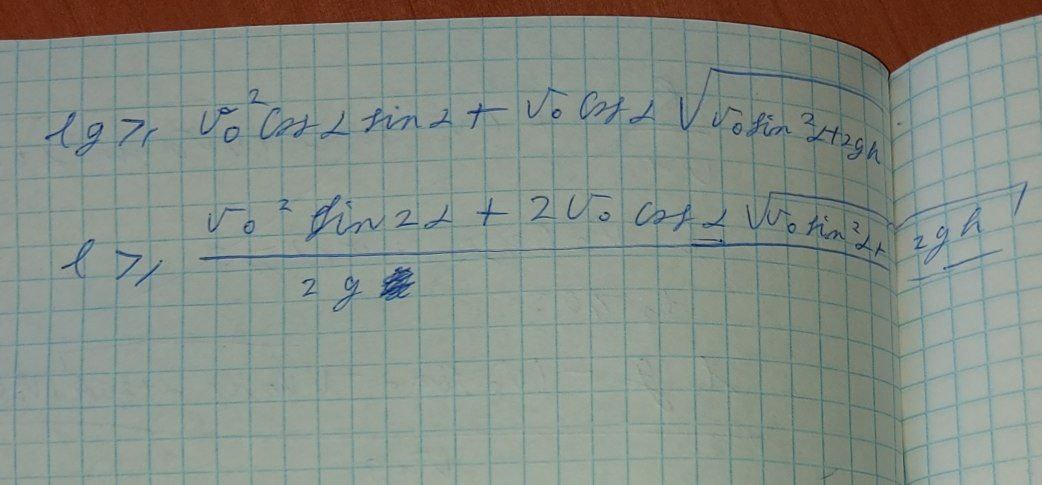
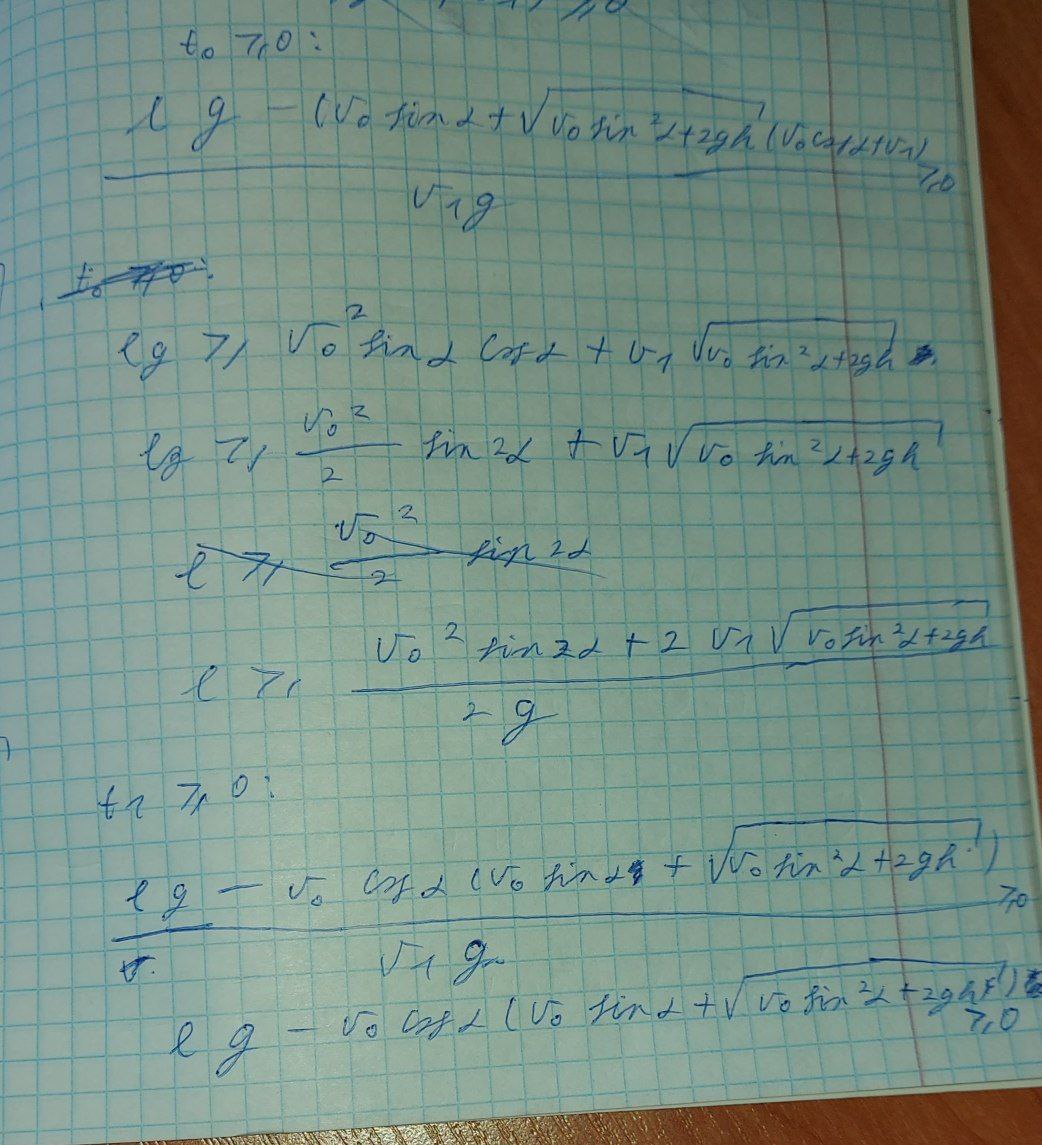
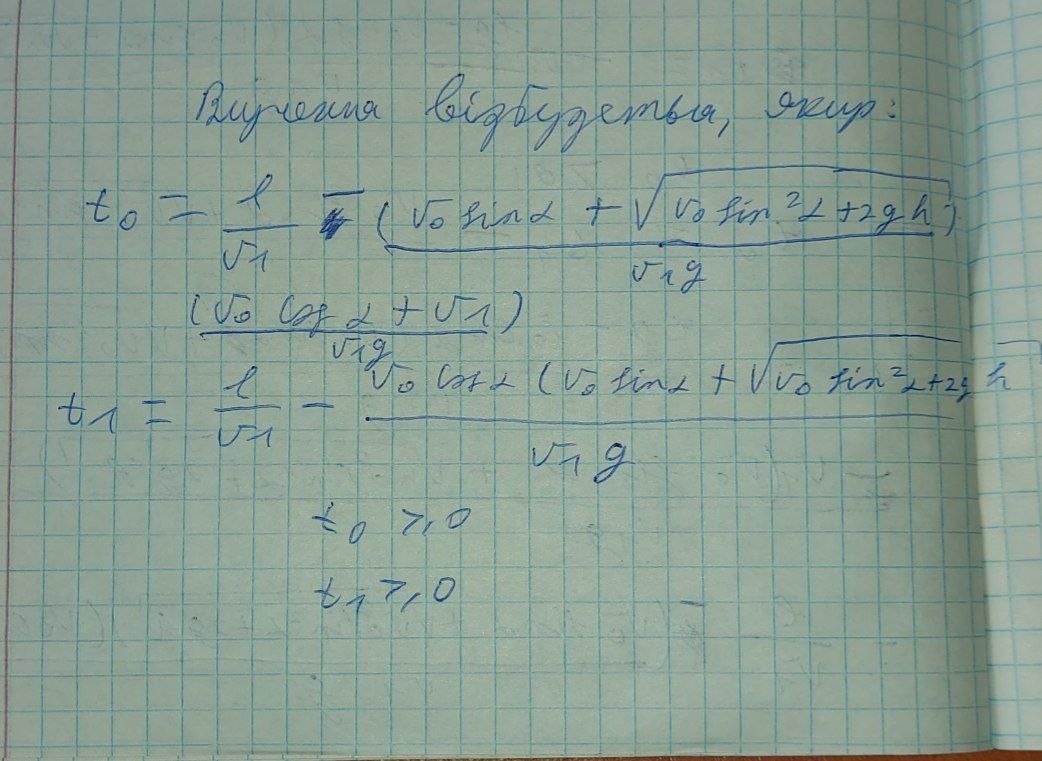
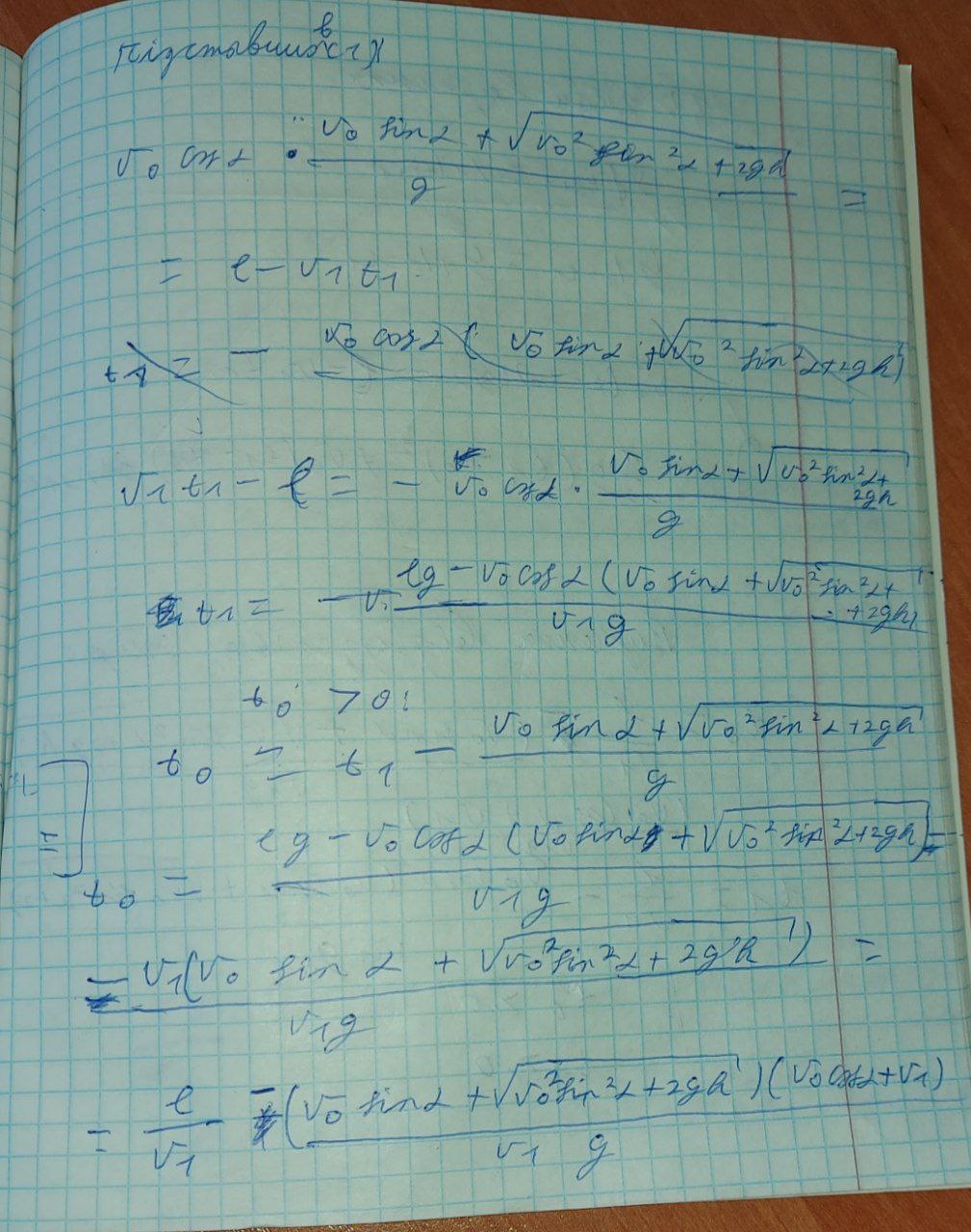
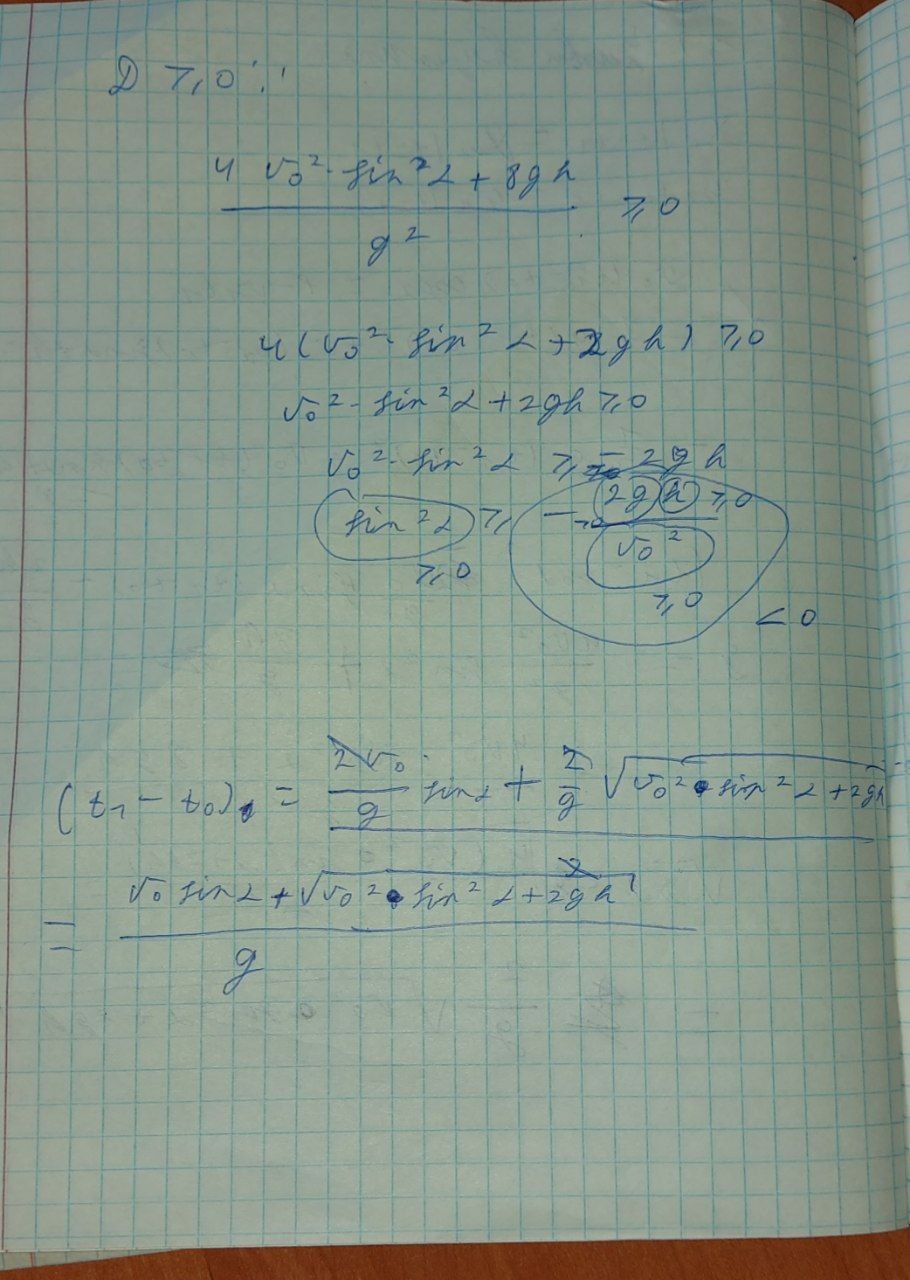






1. **Умови влучання**





1. **Рівняння з опором повітря**

