Державний вищий навчальний заклад

Ужгородський національний університет

Факультет інформаційних технологій

Кафедра програмного забезпечення систем

**ЛАБОРАТОРНА РОБОТА № 1**

**Тема**: **Моделювання руху тіла, кинутого під кутом до горизонту**

Виконав:

студент ІІ курсу

спеціальності: 121 інженерія програмного забезпечення

Вако Сергій Романович

**Ужгород-2024**

**Мета**: Дослідити та візуалізувати рух тіла, кинутого під кутом до горизонту. Використовуючи математичні моделі аналізувати траєкторію руху, висоту максимального підйому, час польоту та інші параметри.

**Короткі теоретичні відомості**

Продовжуючи тематику попередньої роботи, розглянемо рух тіла кинутогопід кутом до горизонту, в полі сили тяжіння.

Так рухається лижник під час стрибка з трампліну, струмінь води з брандспойта, тенісний м’яч під час удару тенісною ракеткою тощо.

  Вивчення особливостей такого руху почалося в XVI столітті і було пов’язане з появою і удосконаленням артилерійських гармат.

  Близько до правильного рішення питання про політ гарматних ядер дійшов італійський математик Нікколо Тарталья, він розглядає питання траєкторії руху снарядів і стверджує, що ця траєкторія є кривою лінією на протязі всього руху, він показує, що найбільшу дальність польоту снарядів можна досягти під час стрільби під кутом 450 до горизонту та формулює правила стріляння, якими артилеристи керувалися до середини XVIІ століття.

Проте повне рішення проблеми, пов’язане з рухом тіл, кинутих під кутом до горизонту, здійснив італійський астроном і фізик Галілео Галілей.

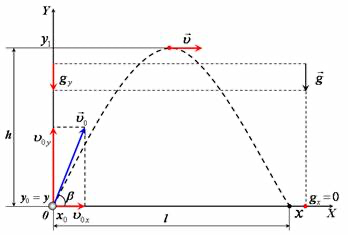
Заслугою Галілео Галілея стало те, що він вперше запропонував розглядати рух тіла, кинутого під кутом до горизонту як **результат складання двох прямолінійних рухів**: *рівномірного руху* по горизонталі і *рівноприскореного* – по вертикалі.

З’явилася нова наука – **балістика.** Балістика – розділ механіки, який вивчає рух тіла під впливом сили тяжіння Землі.

**Рух тіла, кинутого під кутом до горизонту, характеризується:**

* часом польоту (tпол.);
* дальністю польоту (l);
* найбільшою висотою підняття тіла над горизонтом (h).

Знайдемо дальність польоту і максимальну висоту підняття тіла, кинутого під кутом до горизонту. Для цього зобразимо схему руху. Оскільки рух тіла, кинутого під кутом до горизонту відбувається у вертикальній площині, то нам знадобляться дві взаємо перпендикулярні осі *ОХ* та *OY* для його описання.



З’єднаємо початок координат з початковим положенням тіла, спрямуємо вісь ОУ вертикально вгору, а вісь ОХ – горизонтально. У обраній системі координат

,

Запишемо рівняння руху в проекціях на кожну вісь:

**,**

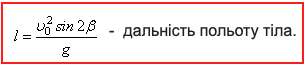
****

Оскільки під час польоту на кинуте тіло діє лише сила земного тяжіння, то тіло, кинуте під кутом до горизонту рухатиметься з прискоренням вільного падіння ***g***. Спроектувавши вектори початкової швидкості  і прискорення   на осі *ОХ* та *OY* (рис 1), запишемо рівняння руху для даного випадку:



звідки ,

підставивши вирази у рівняння для дальності польоту, отримаємо:



Щоб знайти максимальну висоту підняття, запишемо рівняння руху та рівняння швидкості в проекціях на вісь OY, але не для всього часу польоту, а для польоту з точки, ордината якої  в точку з ординатою .

****

Як видно з рисунку в точці з ординатою  проекція миттєвої швидкості тіла на вісь *OY* дорівнює нулю: . Спроектуємо вектори початкової швидкості  і прискорення   на вісь *ОХ* та *OY*, прийнявши до уваги, що :

****

Порівнявши формули можна зробити висновок, що час підйому тіла до найвищої точки траєкторії дорівнює половині часу всього руху тіла.

Підставивши один вираз в інший та виконавши математичні перетворення отримаємо:

****

**Завдання до лабораторної роботи**

1. Визначте математичну модель руху тіла, кинутого під кутом к горизонті, без опору повітря. Використовуйте рівняння руху для горизонтального та вертикального напрямків
2. Використовуючи набуті в попередній роботі навички програмування на мові **JavaScript**, та за допомогою **HTML** розмітки і каскадних таблиць стилів **CSS** побудувати траєкторію руху тіла кинутого під кутом до горизонту
3. Використовуючи попередній результата зробити рефакторинг коду, та замість використання Canvas використати бібліотеку для побудови діаграм та графіків в JavaScript [D3.JS](https://d3js.org)
4. Оформити звіт виконаної роботи, згідно взірця
5. Отримані результати (**лістинг коду** завантажити на **репозиторій**). У звіті потрібно вказати посилання на репозиторій.

**Хід роботи**

**Код:**

import React, { useEffect, useRef, useState } from "react";

import \* as d3 from "d3";

const g = 9.81; // Прискорення вільного падіння

const Home: React.FC = () => {

const svgRef = useRef<SVGSVGElement | null>(null);

const [velocity, setVelocity] = useState(30); // Початкова швидкість (м/с)

const [angle, setAngle] = useState(45); // Кут (градуси)

useEffect(() => {

if (!svgRef.current) return;

const width = 600, height = 400;

const svg = d3.select(svgRef.current);

svg.selectAll("\*").remove(); // Очистка перед побудовою нового графіка

// Перетворюємо кут у радіани

const angleRad = (angle \* Math.PI) / 180;

// Час польоту

const flightTime = (2 \* velocity \* Math.sin(angleRad)) / g;

const dt = 0.05; // Крок ітерації

let points: [number, number][] = [];

for (let t = 0; t <= flightTime; t += dt) {

const x = velocity \* Math.cos(angleRad) \* t;

const y = velocity \* Math.sin(angleRad) \* t - (0.5 \* g \* t \* t);

if (y >= 0) points.push([x, y]);

}

// Масштабування координат

const xScale = d3.scaleLinear()

.domain([0, d3.max(points, d => d[0]) || 1])

.range([50, width - 50]);

const yScale = d3.scaleLinear()

.domain([0, d3.max(points, d => d[1]) || 1])

.range([height - 50, 50]);

// Лінія траєкторії

const line = d3.line<[number, number]>()

.x(d => xScale(d[0]))

.y(d => yScale(d[1]))

.curve(d3.curveMonotoneX);

svg.append("path")

.datum(points)

.attr("fill", "none")

.attr("stroke", "blue")

.attr("stroke-width", 2)

.attr("d", line);

// Вісі

svg.append("line")

.attr("x1", 50).attr("y1", height - 50)

.attr("x2", width - 50).attr("y2", height - 50)

.attr("stroke", "black");

svg.append("line")

.attr("x1", 50).attr("y1", 50)

.attr("x2", 50).attr("y2", height - 50)

.attr("stroke", "black");

}, [velocity, angle]);

return (

<div style={{display: "flex", flexDirection:"column"}}>

<h2>Траєкторія руху тіла</h2>

<label>Початкова швидкість (м/с):</label>

<input style={{width:300}} type="number" value={velocity} onChange={e => setVelocity(+e.target.value)} />

<label>Кут (°):</label>

<input style={{width:300}} type="number" value={angle} onChange={e => setAngle(+e.target.value)} />

<svg ref={svgRef} width={600} height={400} style={{ border: "1px solid black" }}></svg>

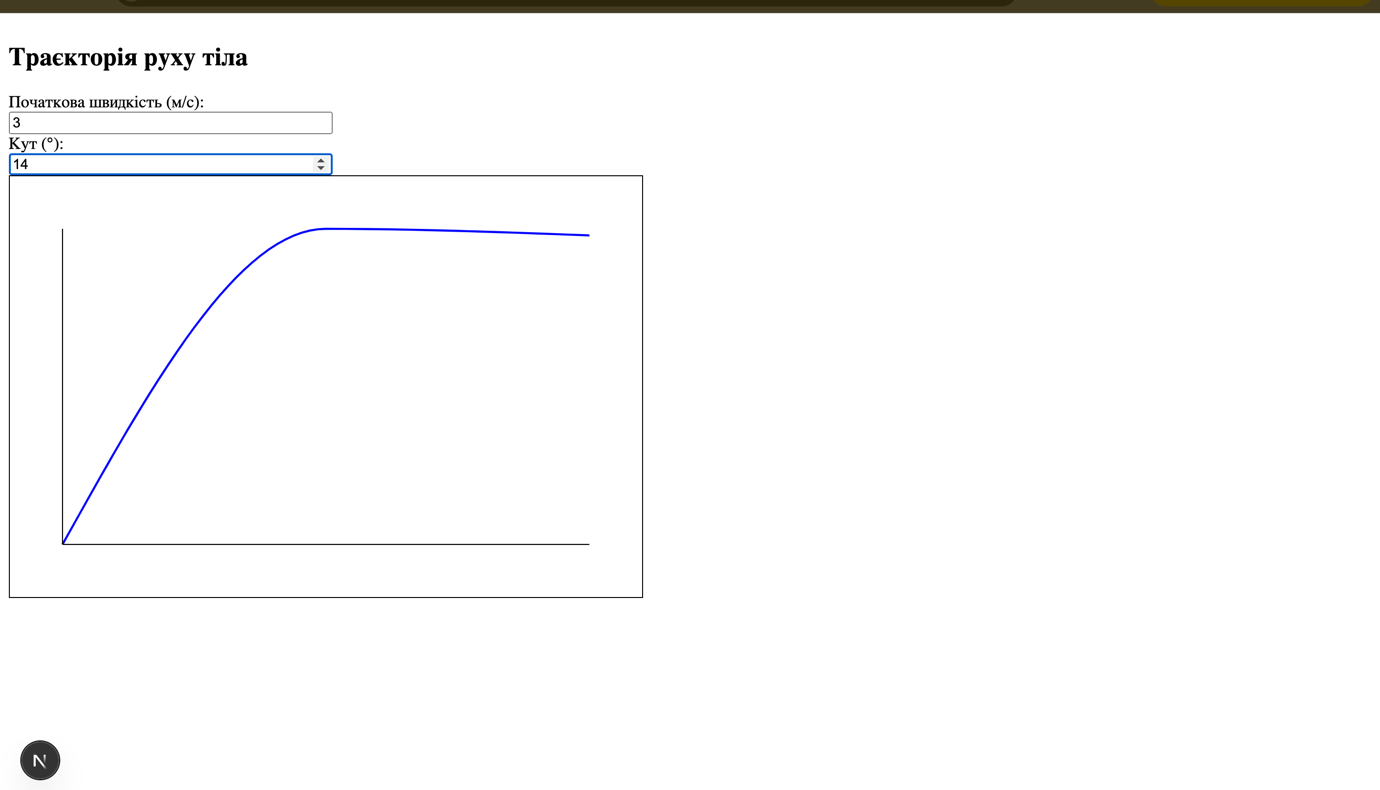
</div>

);

};

export default Home;

**Приклад роботи:**

****

**Висновки:**

Ми розглянули математичну модель руху тіла, кинутого під кутом до горизонту, без врахування опору повітря. Використано рівняння для визначення горизонтальної та вертикальної координат, а також формулу для розрахунку часу польоту. Реалізацію виконано за допомогою TypeScript та React, що забезпечує зручне оновлення UI при зміні параметрів. Для візуалізації використано бібліотеку D3.js, що дозволило створити якісний графік без використання Canvas. Покращення в порівнянні з попередньою реалізацією включають динамічне оновлення траєкторії, масштабування координат та кращу стилізацію. У майбутньому можна додати анімацію руху, врахувати опір повітря або створити 3D-візуалізацію для більш точного моделювання.