**Звіт**

**до лабораторної роботи №3**

Оформив студент групи ТТП-31

Казмірчук Володимир

Викладач: Голубєва Катерина

Постановка задачі

Запрограмувати розв’язання задачі Коші методом Ейлера та Рунге-Кутта четвертого порядку точності:

Розв’язок

Для вирішення цієї задачі було написано наступний код на мові програмування Javascript

const k = 2, ys = -3, x0 = 0, y0 = 0, step = 0.1;

let epoints = [], rpoints = [];

// Draw axes

function axes (ctx, w, h, gx, gy) {

    ctx.beginPath ();

    ctx.strokeStyle = "rgb(0 0 0 / 50%)";

    ctx.lineWidth = 2;

    // X-axis

    ctx.moveTo (0, h / 2);

    ctx.lineTo (w, h / 2);

    // Y-axis

    ctx.moveTo (w / 2, 0);

    ctx.lineTo (w / 2, h);

    ctx.stroke ();

    // Setting style and font

    const fs = 12;

    ctx.font = `${fs}px Arial`;

    ctx.fillStyle = "rgb(0 0 0 / 50%)";

    let step = 1 + Math.round  (gx \* gy / 2000);

    step = step > 5 ? step - step % 5 : 1;

    console.log (step);

    ctx.strokeStyle = "rgb(0 0 0 / 20%)";

    ctx.lineWidth = 1;

    // Grid and labels for X

    const lsx = 0.2 \* w / gx;

    for (let x = -gx; x <= gx; x ++) {

        if (x == 0) { continue; }

        const pos = w / 2 + x \* w / gx;

        ctx.moveTo (pos, h / 2 + lsx);

        ctx.lineTo (pos, h / 2 - lsx);

        ctx.lineTo (pos, h / 2 + lsx);

        ctx.moveTo (pos, 0);

        ctx.lineTo (pos, h);

        if (Math.abs (x) % step == 0) {

            ctx.fillText (`${x}`, pos - fs \* 0.5, h / 2 + fs \* 1.5);

        }

    }

    // Grid and labels for Y

    const lsy = 0.2 \* h / gy;

    for (let y = -gy; y <= gy; y ++) {

        if (y == 0) { continue; }

        const pos = h / 2 + y \* h / gy;

        ctx.moveTo (w / 2 + lsy, pos);

        ctx.lineTo (w / 2 - lsy, pos);

        ctx.lineTo (w / 2 + lsy, pos);

        ctx.moveTo (0, pos);

        ctx.lineTo (w, pos);

        if (Math.abs (y) % step == 0) {

            ctx.fillText (`${-y}`, w / 2 - fs \* 1.5, pos + fs \* 0.5);

        }

    }

    ctx.stroke ();

    ctx.closePath ();

}

// Draw sensors, points and lines

function drawsfunc (ctx, w, h, gx, gy) {

    if (epoints.length > 0) {

        ctx.beginPath ();

        ctx.strokeStyle = "rgb(0 255 0)";

        ctx.lineWidth = 3;

        epoints.forEach (p => {

            const px = w / 2 + p [0] \* w / gx;

            const py = h / 2 - p [1] \* h / gy;

            if (p != epoints [0]) { ctx.lineTo (px, py); }

            ctx.moveTo (px, py);

        });

        ctx.stroke ();

        ctx.closePath ();

    }

    if (rpoints.length > 0) {

        ctx.beginPath ();

        ctx.strokeStyle = "rgb(255 0 0)";

        ctx.lineWidth = 3;

        rpoints.forEach (p => {

            const px = w / 2 + p [0] \* w / gx;

            const py = h / 2 - p [1] \* h / gy;

            if (p != rpoints [0]) { ctx.lineTo (px, py); }

            ctx.moveTo (px, py);

        });

        ctx.stroke ();

        ctx.closePath ();

    }

}

// Display everything

function draw () {

    const canvas = document.querySelector(".Canvas");

    const w = (canvas.width = window.innerWidth);

    const h = (canvas.height = window.innerHeight);

    const ctx = canvas.getContext ("2d");

    ctx.clearRect (0, 0, w, h);

    const gx = Math.round (w / 100);

    const gy = Math.round (h / 100);

    //const gx = 25;

    //const gy = 25;

    console.log (gx, gy);

    axes (ctx, w, h, gx, gy);

    drawsfunc (ctx, w, h, gx, gy);

}

// Derivative of a given equation

function derivative (y) { return -k \* (y - ys); }

// Average derivative for Runge-Kutta method

function avgderivative (y, right) {

    let k0 = derivative (y), k1, k2, k3;

    if (right) {

        k1 = derivative (y + step \* k0 / 2);

        k2 = derivative (y + step \* k1 / 2);

        k3 = derivative (y + step \* k2);

    }

    else {

        k1 = derivative (y - step \* k0 / 2);

        k2 = derivative (y - step \* k1 / 2);

        k3 = derivative (y - step \* k2);

    }

    return (k0 + 2 \* k1 + 2 \* k2 + k3) / 6;

}

// Euler method

function euler (from, to) {

    epoints = [[x0, y0]];

    for (let x = x0, y = y0; x <= to + step; x += step, y += derivative (y) \* step) {

        epoints.push ([x, y]);

    }

    for (let x = x0, y = y0; x >= from - step; x -= step, y -= derivative (y) \* step) {

        epoints.unshift ([x, y]);

    }

}

// Runge-Kutta method

function runge (from, to) {

    rpoints = [[x0, y0]];

    for (let x = x0, y = y0; x <= to + step; x += step, y += avgderivative (y, true) \* step) {

        rpoints.push ([x, y]);

    }

    for (let x = x0, y = y0; x >= from - step; x -= step, y -= avgderivative (y, false) \* step) {

        rpoints.unshift ([x, y]);

    }

}

// Calculate and draw

euler (-1, 10);

runge (-1, 10);

addEventListener ("resize", (event) => {});

onresize = (event) => draw ();

draw ();

Цей код реалізує числові методи (метод Ейлера та метод Рунге-Кутта) для розв'язання диференціального рівняння та візуалізацію отриманих результатів. Загалом, скрипт виконує такі дії:

**1. Ініціалізація даних:**

- Задаються константи *k*, *ys*, *x0*, *y0*, та *step*.

- Створюються порожні масиви *epoints* та *rpoints* для зберігання точок розв'язку методом Ейлера та методом Рунге-Кутта відповідно.

**2. Функція виведення вісей та сітки на графіку:**

- Функція *axes* рисує координатні вісі та сітку на графіку.

**3. Функція виведення ліній на графіку:**

- Функція *drawsfunc* відображає лінії, що представляють результати методу Ейлера та методу Рунге-Кутта.

**4. Функція виведення результатів на екран:**

- Функція *draw* викликається для відображення результатів на графіку при зміні розміру вікна.

**5. Функції для числових методів:**

- Функція *derivative* обчислює похідну від заданого диференціального рівняння.

- Функція *avgderivative* обчислює середню похідну для методу Рунге-Кутта.

- Функції *euler* та *runge* використовують методи Ейлера та Рунге-Кутта для розв'язання диференціального рівняння відповідно.

**6. Обчислення та виведення результатів:**

- Виконується обчислення методами Ейлера та Рунге-Кутта для вказаного діапазону.

- Виводяться результати на екран.

**7. Обробка події *resize*: та виклик функції *draw*:**

- Визначена функція, яка викликається при зміні розміру вікна для перерисовки графіку.

- Забезпечує перше відображення результатів при завантаженні сторінки.

Загалом, цей код може бути використаний для візуалізації та порівняння різних числових методів для розв'язання диференціальних рівнянь, чого і вимагає поставлена задача.

Приклади роботи програми

Приклад №1

A graph of a function

Description automatically generated

Зеленим зображений графік рівняння для методу Ейлера, червоним – методу Рунге-Кутта. Параметри: k = 2, ys = -3, x0 = 0, y0 = 0, функцію обчислено в інтервалі [-1; 10] проміжками в 0.1.