НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №3

із дисципліни «Розподілені і хмарні обчислення»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-01 | *Доцент кафедри ПМА*  *Ліскін В. О.* |
| *Романецький М.С.* |  |

Київ — 2023

Мета роботи: Проінтегрувати методом редукії. Проінтегрувати методом Сімпсона за різної кількості потоків та інтервалів розбиття.

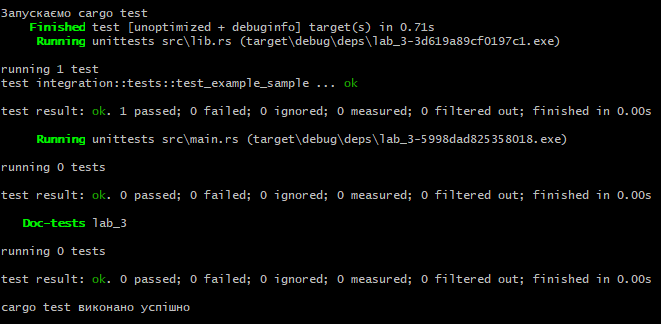
Опис програми: Бібліотека ‘Rayon’ буде використовуватись для паралелізму.

Для порівняння часу роботи буде використовувати ‘Criterion’

|  |  |  |
| --- | --- | --- |
| Інтервали розбиття | Потоки | Середній час виконання |
| 10 | 1 | 3.8007 мкс |
| 10 | 2 | 6.4917 мкс |
| 10 | 4 | 2.7270 мкс |
| 10 | 8 | 4.2251 мкс |
|  | 1 | 257.71 мкс |
|  | 2 | 154.58 мкс |
|  | 4 | 38.784 мкс |
|  | 8 | 34.539 мкс |
|  | 1 | 285.08 мс |
|  | 2 | 146.75 мс |
|  | 4 | 46.162 мс |
|  | 8 | 18.695 мс |



Завдання 1



Тестування



Інтерфейс

Код програми:

*// cd /d/KPI/Distributed\_computing/Labs/lab\_3*

use std::{sync::Arc, f64::consts::E};

use lab\_3::integration::integral\_reduction;

use eframe::{epi, egui::{*self*, CtxRef}};

use std::process::Command;

use egui::vec2;

struct MyApp {

*//  дані та стан програми*

}

impl Default for MyApp {

    fn default() -> *Self* {

*Self* {

*// Ініціалізація стану*

        }

    }

}

impl epi::App for MyApp {

    fn name(&*self*) -> &str {

        "Romanetskiy | Lab 3"

    }

    fn update(&mut *self*, ctx: &CtxRef, \_frame: &mut epi::Frame) {

        egui::CentralPanel::default().show(ctx, |ui| {

            let button\_size = vec2(ui.available\_width(), 95.0);

            if ui.add\_sized(button\_size, egui::Button::new("Run Task 1")).clicked() {

                task\_1()

            }

            if ui.add\_sized(button\_size, egui::Button::new("Run Task 2")).clicked() {

                task\_2()

            }

            if ui.add\_sized(button\_size, egui::Button::new("Run Test")).clicked() {

                testing()

            }

        });

    }

}

fn task\_1() {

    println!("\nTask 1:");

    let f = |x: f64| (E.powf(x) - 1.0) / (E.powf(x) + 1.0);

    let a = 1.0;

    let b = 2.0;

    let steps = 100;

    println!("f = (e^x-1) / (e^x+1), a={}, b={}, steps={}", a, b, steps);

    let af = Arc::new(f);

    let res = integral\_reduction(&af, a, b, steps);

    println!("Result: {}", res);

}

fn task\_2() {

    println!("\nTask 2:");

    println!("Запускаємо cargo bench");

    match Command::new("cargo").args(&["bench"]).status() {

        Ok(status) => if status.success() {

            println!("cargo bench виконано успішно");

        } else {

            eprintln!("cargo bench завершилося з помилкою");

        },

        Err(e) => eprintln!("Помилка при запуску cargo bench: {}", e),

    }

}

fn testing() {

    println!("\nЗапускаємо cargo test");

    match Command::new("cargo").args(&["test"]).status() {

        Ok(status) => if status.success() {

            println!("cargo test виконано успішно");

        } else {

            eprintln!("cargo test завершилося з помилкою");

        },

        Err(e) => eprintln!("Помилка при запуску cargo test: {}", e),

    }

}

fn main() {

    let app = MyApp::default();

    let mut native\_options = eframe::NativeOptions::default();

    native\_options.initial\_window\_size = Some(egui::vec2(400.0, 300.0));

    eframe::run\_native(Box::new(app), native\_options);

}