НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №5

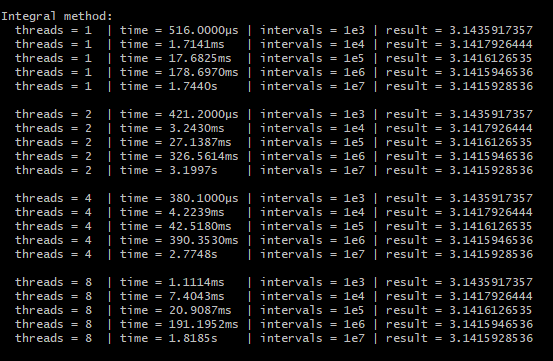
із дисципліни «Розподілені і хмарні обчислення»

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

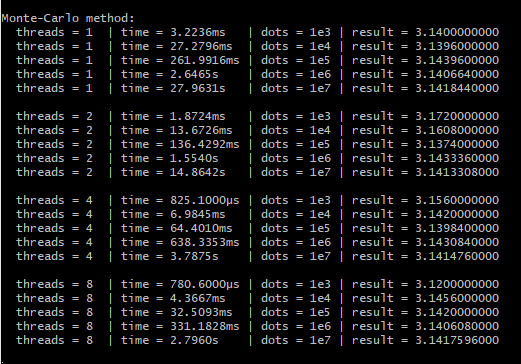
Київ — 2023

Мета роботи: Розпаралелити метод обчислення константи PI

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

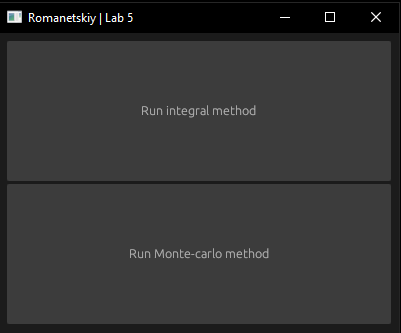


Інтегральний метод



Метод Монте-Карло

Висновки: Інтегральний метод виявився точнішим за Монте-Карло



Інтерфейс

Код програми:

*// cd D:/KPI/Distributed\_computing/Labs/lab\_5*

use egui::vec2;

use std::time::Instant;

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

use lab\_5::constcalc::{picalc, mc\_picalc};

use rayon::ThreadPoolBuilder;

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 5"

    }

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

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

            let button\_size = vec2(ui.available\_width(), 140.0);

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

                task\_intergral()

            }

            if ui.add\_sized(button\_size, egui::Button::new("Run Monte-carlo method")).clicked() {

                task\_monte\_carlo()

            }

        });

    }

}

fn format\_dots(n: i32) -> String {

    let mut dots = n.to\_string();

    let len = dots.len();

    if len > 3 {

        let e = len - 1;

        dots = format!("1e{}", e);

    }

    dots

}

fn task\_intergral() {

    println!("\nIntegral method:");

    for &threads in [1, 2, 4, 8].iter() {

        let pool = ThreadPoolBuilder::new().num\_threads(threads).build().unwrap();

        for &n in [1e3 as i32, 1e4 as i32, 1e5 as i32, 1e6 as i32, 1e7 as i32].iter() {

            let start = Instant::now();

            pool.install(|| {

                let pi = picalc(n);

                let duration = start.elapsed();

                println!("  threads = {:<2} | time = {:<10} | intervals = {:<2} | result = {:.10}",

                        threads, format!("{:.4?}", duration), format\_dots(n), pi);

            });

        }

        println!();

    }

}

fn task\_monte\_carlo() {

    println!("\n\n\nMonte-Carlo method:");

    for &threads in [1, 2, 4, 8].iter() {

        let pool = ThreadPoolBuilder::new().num\_threads(threads).build().unwrap();

        for &n in [1e3 as i32, 1e4 as i32, 1e5 as i32, 1e6 as i32, 1e7 as i32].iter() {

            let start = Instant::now();

            pool.install(|| {

                let pi = mc\_picalc(n);

                let duration = start.elapsed();

                println!("  threads = {:<2} | time = {:<10} | dots = {:<2} | result = {:.10}",

                        threads, format!("{:.4?}", duration), format\_dots(n), pi);

            });

        }

        println!();

    }

}

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);

}