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:

студент групи КМ-01

Романецький М.С.

Керівник:

Доцент кафедри ПМА
Ліскін В. О.

Київ — 2023

Мета роботи: Розпаралелити метод обчислення константи PI
Опис програми: Бібліотека ‘Rayon’ буде використовуватись для паралелізму.

```
Integral method:
  threads = 1  | time = 516.0000µs | intervals = 1e3 | result = 3.1435917357
  threads = 1  | time = 1.7141ms   | intervals = 1e4 | result = 3.1417926444
  threads = 1  | time = 17.6825ms  | intervals = 1e5 | result = 3.1416126535
  threads = 1  | time = 178.6970ms | intervals = 1e6 | result = 3.1415946536
  threads = 1  | time = 1.7440s    | intervals = 1e7 | result = 3.1415928536

  threads = 2  | time = 421.2000µs | intervals = 1e3 | result = 3.1435917357
  threads = 2  | time = 3.2430ms   | intervals = 1e4 | result = 3.1417926444
  threads = 2  | time = 27.1387ms  | intervals = 1e5 | result = 3.1416126535
  threads = 2  | time = 326.5614ms | intervals = 1e6 | result = 3.1415946536
  threads = 2  | time = 3.1997s    | intervals = 1e7 | result = 3.1415928536

  threads = 4  | time = 380.1000µs | intervals = 1e3 | result = 3.1435917357
  threads = 4  | time = 4.2239ms   | intervals = 1e4 | result = 3.1417926444
  threads = 4  | time = 42.5180ms  | intervals = 1e5 | result = 3.1416126535
  threads = 4  | time = 390.3530ms | intervals = 1e6 | result = 3.1415946536
  threads = 4  | time = 2.7748s    | intervals = 1e7 | result = 3.1415928536

  threads = 8  | time = 1.1114ms   | intervals = 1e3 | result = 3.1435917357
  threads = 8  | time = 7.4043ms   | intervals = 1e4 | result = 3.1417926444
  threads = 8  | time = 20.9087ms  | intervals = 1e5 | result = 3.1416126535
  threads = 8  | time = 191.1952ms | intervals = 1e6 | result = 3.1415946536
  threads = 8  | time = 1.8185s    | intervals = 1e7 | result = 3.1415928536
```

Інтегральний метод

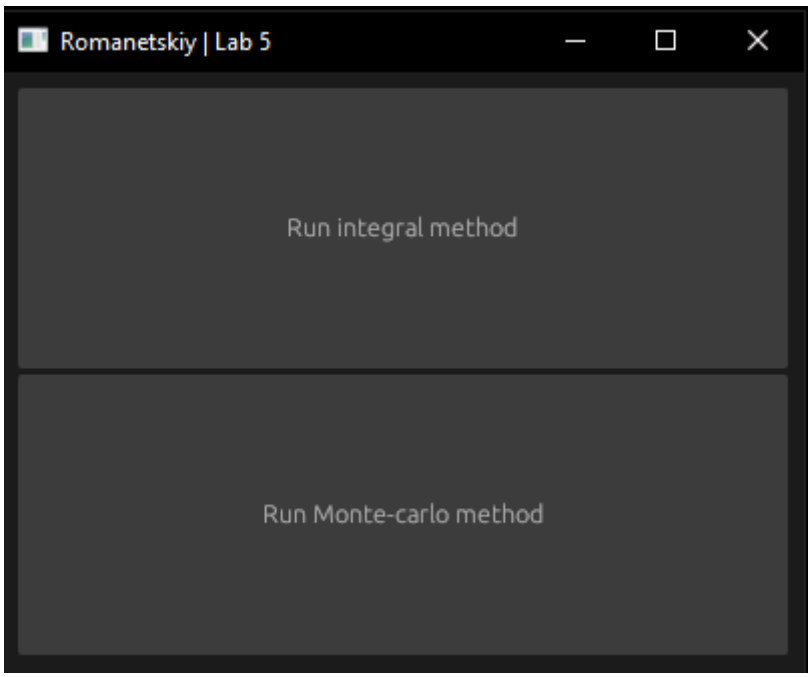
```
Monte-Carlo method:
  threads = 1  | time = 3.2236ms   | dots = 1e3 | result = 3.1400000000
  threads = 1  | time = 27.2796ms  | dots = 1e4 | result = 3.1396000000
  threads = 1  | time = 261.9916ms | dots = 1e5 | result = 3.1439600000
  threads = 1  | time = 2.6465s    | dots = 1e6 | result = 3.1406640000
  threads = 1  | time = 27.9631s   | dots = 1e7 | result = 3.1418440000

  threads = 2  | time = 1.8724ms   | dots = 1e3 | result = 3.1720000000
  threads = 2  | time = 13.6726ms  | dots = 1e4 | result = 3.1608000000
  threads = 2  | time = 136.4292ms | dots = 1e5 | result = 3.1374000000
  threads = 2  | time = 1.5540s    | dots = 1e6 | result = 3.1433360000
  threads = 2  | time = 14.8642s   | dots = 1e7 | result = 3.1413308000

  threads = 4  | time = 825.1000µs | dots = 1e3 | result = 3.1560000000
  threads = 4  | time = 6.9845ms   | dots = 1e4 | result = 3.1420000000
  threads = 4  | time = 64.4010ms  | dots = 1e5 | result = 3.1398400000
  threads = 4  | time = 638.3353ms | dots = 1e6 | result = 3.1430840000
  threads = 4  | time = 3.7875s    | dots = 1e7 | result = 3.1414760000

  threads = 8  | time = 780.6000µs | dots = 1e3 | result = 3.1200000000
  threads = 8  | time = 4.3667ms   | dots = 1e4 | result = 3.1456000000
  threads = 8  | time = 32.5093ms  | dots = 1e5 | result = 3.1420000000
  threads = 8  | time = 331.1828ms | dots = 1e6 | result = 3.1406080000
  threads = 8  | time = 2.7960s    | dots = 1e7 | result = 3.1417596000
```

Метод Монте-Карло
Висновки: Інтегральний метод виявився точнішим за Монте-Карло



Інтерфейс

Код програми:

```rust
// 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);
}
```