Tugas 3: Which One is Fastest? Julia Optimization: Laplace Toy Problem

```
md""" # Tugas 3: Which One is Fastest? Julia Optimization: Laplace Toy Problem"""
```

Ajam Jamaludin
19/445602/PA/19426
https://ajamj.github.io

Pakettt

```
using BenchmarkTools
```

```
using .Threads
```

```
using Distributed
```

```
using Plots, PlutoUI, Images, Downloads
```

Table of Contents


```
TableOfContents()
```

Pendefinisian Fungsi-Fungsi

lap2d! (generic function with 1 method)

```
# Fungsi Laplace diskrit tanpa optimasi
function lap2d!(u, unew)
    M, N = size(u)
    for j in 2:N-1
        for i in 2:M-1
            unew[i, j] = 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
        end
    end
end
```

inbounds_lap2d! (generic function with 1 method)

```
# Fungsi Laplace diskrit dengan @inbounds
function inbounds_lap2d!(u, unew)
    M, N = size(u)
    for j in 2:N-1
        for i in 2:M-1
            @inbounds unew[i, j] = 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
        end
    end
end
```

is_lap2d! (generic function with 1 method)

```
# Fungsi Laplace diskrit dengan @inbounds dan @simd
function is_lap2d!(u, unew)
    M, N = size(u)
    @inbounds for j in 2:N-1
        @simd for i in 2:M-1
            unew[i, j] = 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
        end
    end
end
```

threaded_lap2d! (generic function with 1 method)

```
# Fungsi Laplace diskrit dengan @threads dan @inbounds
function threaded_lap2d!(u, unew)
    M, N = size(u)
    @threads for j in 2:N-1
        for i in 2:M-1
            @inbounds unew[i, j] = 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
        end
    end
end
```

```
threaded_unbounded_lap2d! (generic function with 1 method)
```

```
# Fungsi Laplace diskrit dengan @threads tanpa @inbounds
function threaded_unbounded_lap2d!(u, unew)
    M, N = size(u)
    @threads for j in 2:N-1
        for i in 2:M-1
            unew[i, j] = 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
        end
    end
end
```

```
threaded_workaround_lap2d! (generic function with 1 method)
```

```
function threaded_workaround_lap2d!(u, unew)
    M, N = size(u)
    partial = zeros(eltype(u), nthreads())

    @threads for j in 2:N-1
        for i in 2:M-1
            partial[threadid()] += 0.25 * (u[i+1, j] + u[i-1, j] + u[i, j+1] + u[i, j-1])
        end
    end

    s = zero(eltype(u))
    for i in 1:nthreads()
        s += partial[i]
    end

    @threads for j in 2:N-1
        for i in 2:M-1
            unew[i, j] = s
        end
    end
end

```

Pendefinisian Matriks dan Variabel

```
256×256 Matrix{Float64}:
 10.0  10.0  10.0  10.0  10.0  10.0  10.0  …  10.0  10.0  10.0  10.0  10.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
  ⋮                          ⋮          ⋱    ⋮                          ⋮
 10.0   0.0   0.0   0.0   0.0   0.0   0.0  …   0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0   0.0   0.0   0.0   0.0   0.0   0.0      0.0   0.0   0.0   0.0   0.0  10.0
 10.0  10.0  10.0  10.0  10.0  10.0  10.0  …  10.0  10.0  10.0  10.0  10.0  10.0
```

```
begin
    # Inisialisasi ukuran matriks
    M = 256
    N = 256
    u = zeros(M,N)
    u[1,:] = u[end,:] = u[:,1] = u[:,end] .= 10.0
    unew = copy(u)
end
```

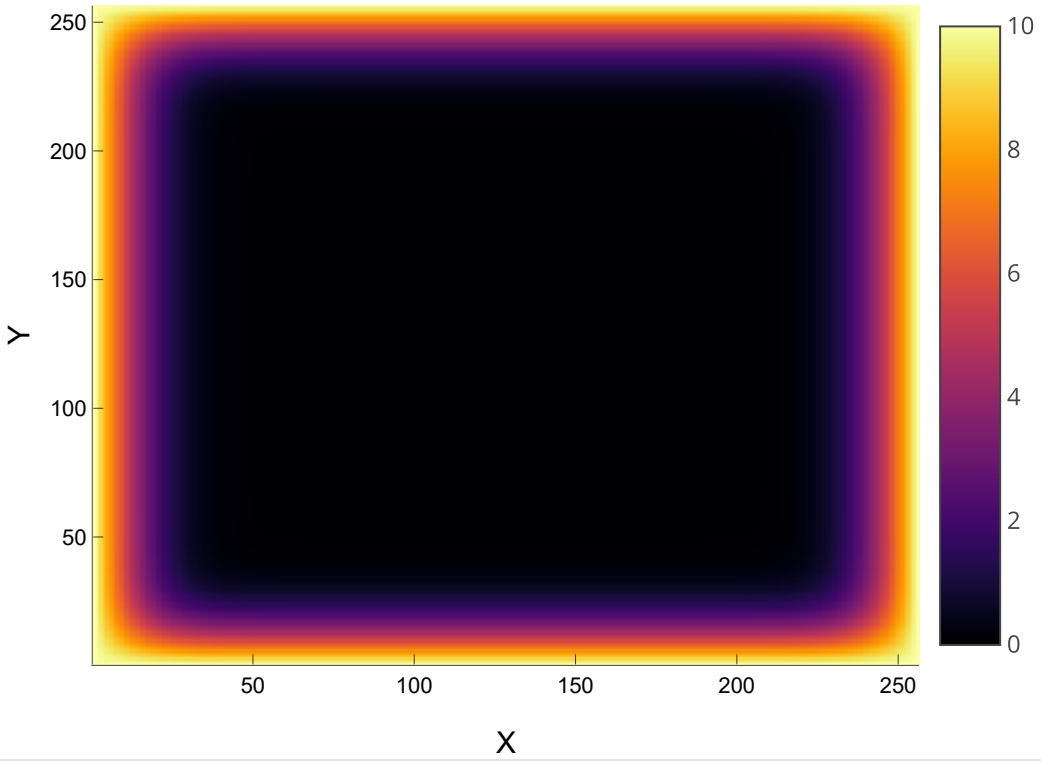
```
maximum_iteration = 500
```

```
maximum_iteration = 500
```

Eksekusi

lap2d

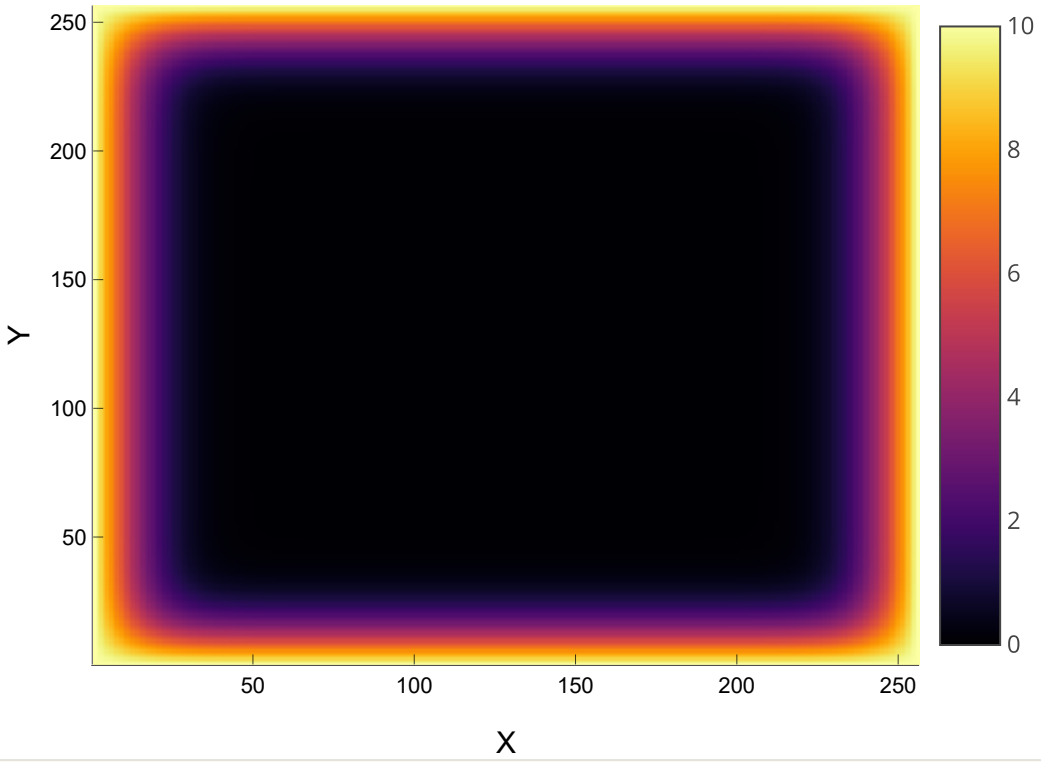
```
for i in 1:maximum_iteration
    lap2d!(u, unew)
    # copy new computed field to old array
    u = copy(unew)
    mt_lap2d = deepcopy(unew)
end
```




```
heatmap(unew,xlabel="X",ylabel="Y")
```

inbounds_lap2d

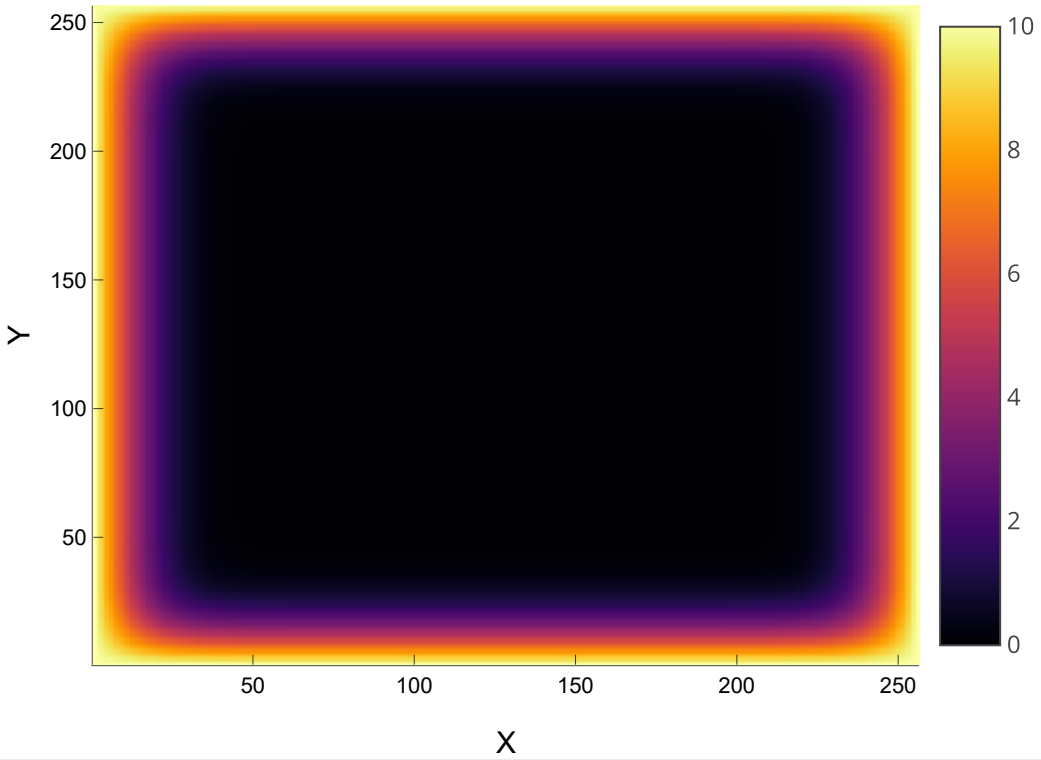
```
for i in 1:maximum_iteration
    inbounds_lap2d!(u, unew)
    # copy new computed field to old array
    u = copy(unew)
end
```




```
heatmap(unew,xlabel="X",ylabel="Y")
```

threaded_lap2d

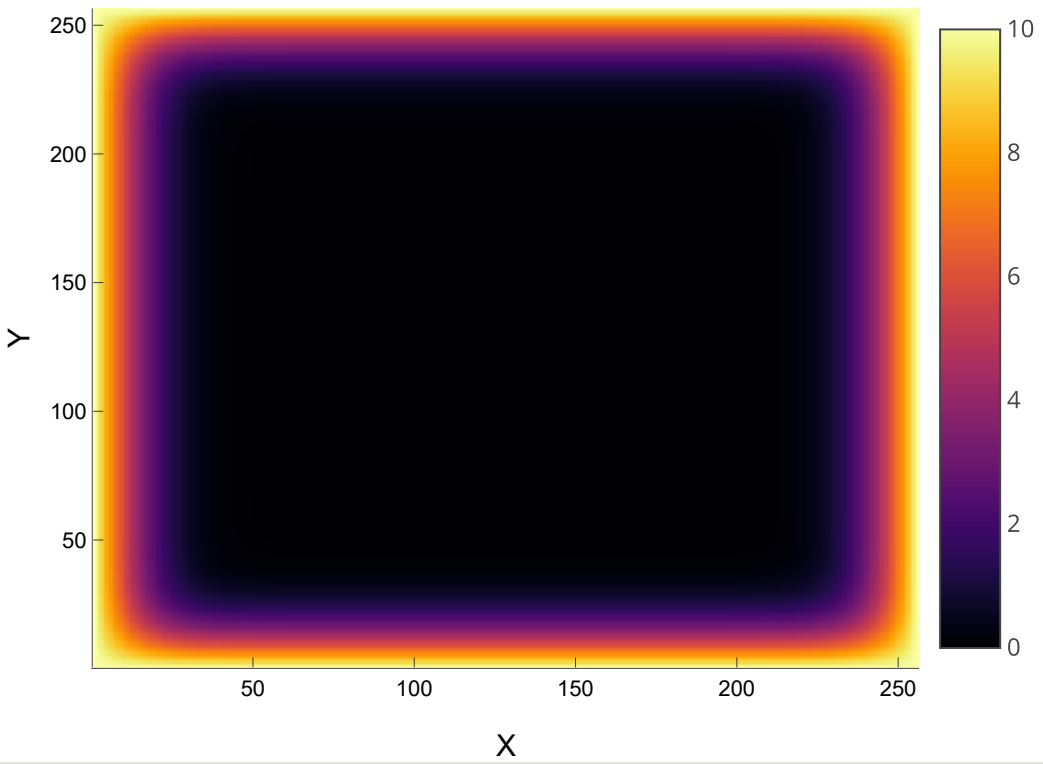
```
for i in 1:maximum_iteration
    threaded_lap2d!(u, unew)
    # copy new computed field to old array
    u = copy(unew)
    mt_tl = deepcopy(unew)
end
```




```
heatmap(unew,xlabel="X",ylabel="Y")
```

is_lap2d

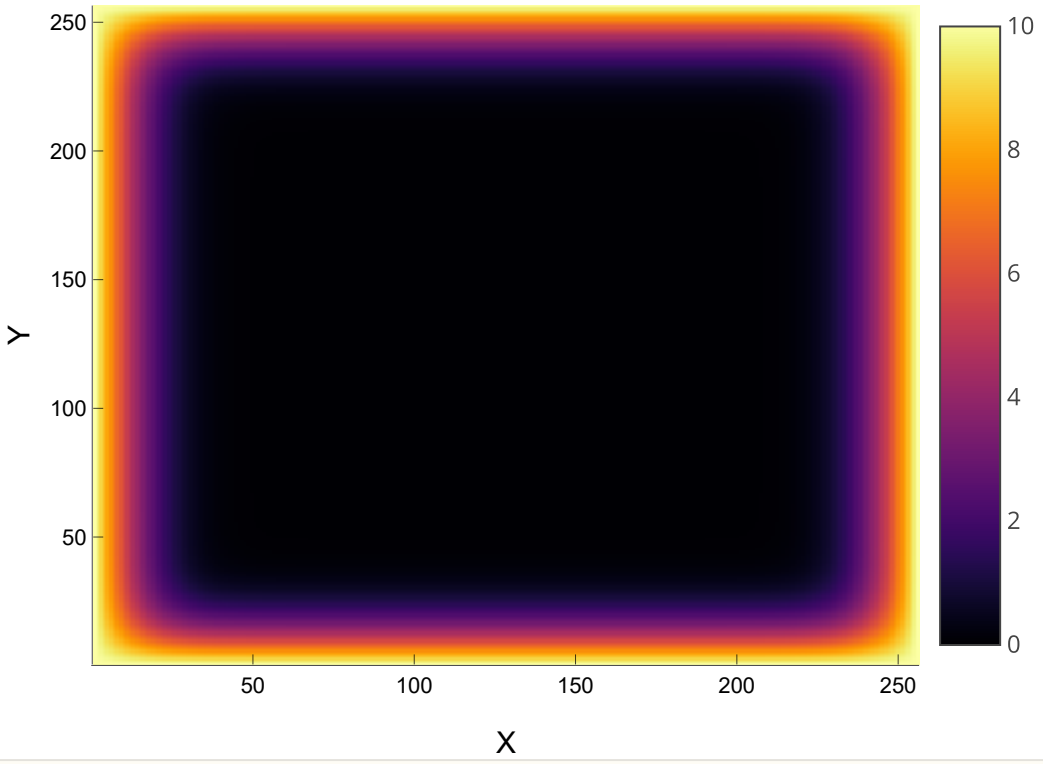
```
for i in 1:maximum_iteration
    is_lap2d!(u, unew)
    # copy new computed field to old array
    u = copy(unew)
end
```




```
heatmap(unew,xlabel="X",ylabel="Y")
```

threaded_unbounded_lap2d

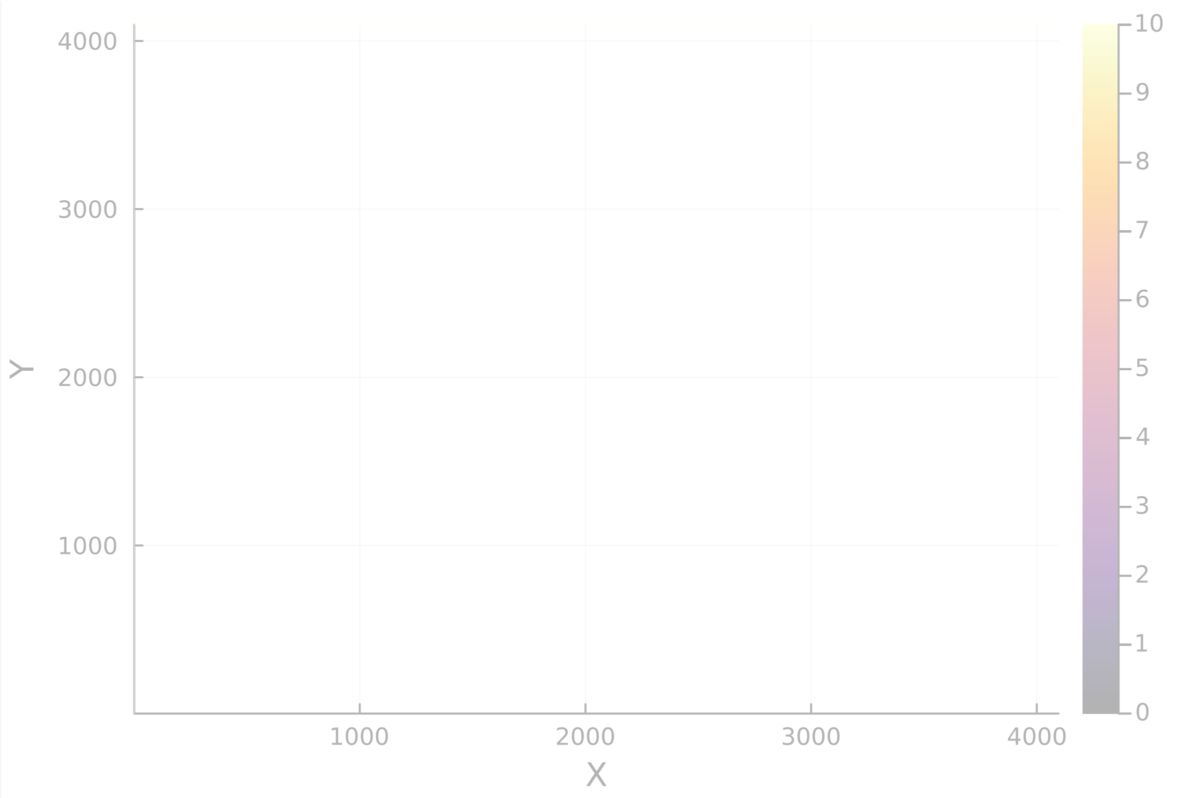
```
for i in 1:maximum_iteration
    threaded_unbounded_lap2d!(u, unew)
    # copy new computed field to old array
    u = copy(unew)
end
```




```
heatmap(unew,xlabel="X",ylabel="Y")
```

threaded_workaround_lap2d!

```
for i in 1:maximum_iteration
    threaded_workaround_lap2d!(u, unew)
    # copy new computed field to old array
    u = copy(unew)
    mt_twl = deepcopy(unew)
end
```




```
heatmap(unew,xlabel="X",ylabel="Y") #ERROR
```

```
▶ PlotlyBackend()
```

```
plotly()
```

Benchmark

```
lap = BenchmarkTools.Trial: 10000 samples with 1 evaluation.
 Range (min … max):  114.500 μs …  11.003 ms  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     129.600 μs               ┊ GC (median):    0.00%
 Time  (mean ± σ):   198.629 μs ± 278.982 μs  ┊ GC (mean ± σ):  0.00% ± 0.00%
```

```
  114 μs        Histogram: log(frequency) by time       1.57 ms <

 Memory estimate: 0 bytes, allocs estimate: 0.
```

```
lap = @benchmark lap2d!($u, $unew)
```

```
inb_lap = BenchmarkTools.Trial: 10000 samples with 1 evaluation.
 Range (min … max):  38.400 μs …  15.621 ms  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     48.800 μs               ┊ GC (median):    0.00%
 Time  (mean ± σ):   83.723 μs ± 215.796 μs  ┊ GC (mean ± σ):  0.00% ± 0.00%

 38.4 μs       Histogram: log(frequency) by time       634 μs <

 Memory estimate: 0 bytes, allocs estimate: 0.
```

```
1 inb_lap = @benchmark inbounds_lap2d!($u, $unew)
```

```
is_lap = BenchmarkTools.Trial: 10000 samples with 1 evaluation.
 Range (min … max):  38.000 μs …   5.168 ms  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     54.200 μs               ┊ GC (median):    0.00%
 Time  (mean ± σ):   94.128 μs ± 177.939 μs  ┊ GC (mean ± σ):  0.00% ± 0.00%

 38 μs         Histogram: log(frequency) by time       789 μs <

 Memory estimate: 0 bytes, allocs estimate: 0.
```

```
1 is_lap = @benchmark is_lap2d!($u, $unew)
```

```
t_lap = BenchmarkTools.Trial: 10000 samples with 1 evaluation.
 Range (min … max):  23.400 μs …   8.169 ms  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     43.900 μs               ┊ GC (median):    0.00%
 Time  (mean ± σ):   91.704 μs ± 229.128 μs  ┊ GC (mean ± σ):  0.00% ± 0.00%

 23.4 μs       Histogram: log(frequency) by time       936 μs <

 Memory estimate: 1.38 KiB, allocs estimate: 12.
```

```
1 t_lap = @benchmark threaded_lap2d!($u, $unew)
```

```
tu_lap = BenchmarkTools.Trial: 10000 samples with 1 evaluation.
 Range (min … max):   74.100 μs …   8.447 ms  ┊ GC (min … max): 0.00% … 0.00%
 Time  (median):     147.500 μs               ┊ GC (median):    0.00%
 Time  (mean ± σ):   214.650 μs ± 345.653 μs  ┊ GC (mean ± σ):  0.00% ± 0.00%

 74.1 μs       Histogram: log(frequency) by time       2.17 ms <

 Memory estimate: 1.38 KiB, allocs estimate: 12.
```

```
1 tu_lap = @benchmark threaded_unbounded_lap2d!($u, $unew)
```

```
0.0741
```

```
1 begin
2     # Mengambil nilai waktu eksekusi terbaik
3     bench_lap = minimum(lap.times)/1e6
4     bench_inb_lap = minimum(inb_lap.times)/1e6
5     bench_is_lap = minimum(is_lap.times)/1e6
6     bench_t_lap = minimum(t_lap.times)/1e6
7     bench_tu_lap = minimum(tu_lap.times)/1e6
8 end
```

```
Dict("inbounds_lap2d" => 0.0384, "is_lap2d" => 0.038, "threaded_unbounded_lap2d" => 0.0
```

```
begin
    # Menampilkan hasil
    results_dict = Dict(
    "lap2d" => bench_lap,
    "inbounds_lap2d" => bench_inb_lap,
    "is_lap2d" => bench_is_lap,
    "threaded_lap2d" => bench_t_lap,
    "threaded_unbounded_lap2d" => bench_tu_lap,
)
end
```

```
sorted_results =
["threaded_lap2d" => 0.0234, "is_lap2d" => 0.038, "inbounds_lap2d" => 0.0384, "threaded.
```

```
sorted_results = sort(collect(results_dict), by=x->x[2])

```

```
for (index, (key, value)) in enumerate(sorted_results)
    println("[$index] $key: $value s")
end

```

```
[1] threaded_lap2d: 0.0234 s
[2] is_lap2d: 0.038 s
[3] inbounds_lap2d: 0.0384 s
[4] threaded_unbounded_lap2d: 0.0741 s
[5] lap2d: 0.1145 s
```

Pembahasan

Dengan ukuran matriks MxN sebesar 512x512, fungsi `is_lap2d` memiliki kecepatan yang paling cepat, yaitu sekitar **0.243 s**. Fungsi yang paling lambat adalah fungsi tanpa optimisasi, `lap2d`, yaitu selama **0.4213 s**.

```
[1] is_lap2d: 0.2403 s
[2] inbounds_lap2d: 0.2498 s
[3] threaded_unbounded_lap2d: 0.3292 s
[4] threaded_lap2d: 0.3825 s
[5] lap2d: 0.4213 s
```

Saat ukuran matriks diperkecil menjadi 256x256, didapat urutan kecepatan sebagai berikut:

```
[1] threaded_lap2d: 0.0251 s
[2] is_lap2d: 0.0413 s
[3] inbounds_lap2d: 0.0416 s
[4] threaded_unbounded_lap2d: 0.0742 s
[5] lap2d: 0.0935 s
```

Fungsi `is_lap2d` yang tadinya di peringkat pertama kini menjadi peringkat ke-dua dengan waktu eksekusi sebesar **0.0251 s**. Nilai tersebut hampir 6 kali lebih cepat dari nilai semula. Fungsi yang lain sama, semuanya hampir meningkat di kisaran 3 sampai 4 kali lipat.

```
[1] threaded_unbounded_lap2d: 1.8123 s
[2] threaded_lap2d: 1.8737 s
[3] is_lap2d: 1.9632 s
[4] inbounds_lap2d: 1.9684 s
[5] lap2d: 2.4211 s
```

Saat matriks diperbesar menjad 1024x1024, peringkat pertama diduduki oleh fungsi `threaded_unbounded_lap2d` dengan waktu eksekusi **1.8123 s**. Fungsi `is_lap2d` turun menjadi peringkat ketiga dengan waktu eksekusi **1.9632 s**. Sementara itu, fungsi `lap2d` tanpa optimisasi masih duduk di peringkat terakhir.

Adapun potensi untuk terjadinya race condition pada fungsi yang multithreading, sepertinya tidak terjadi dalam kasus ini bila dilihat pada hasil plotnya yang sama dengan fungsi tanpa multithreading. Hal tersebut karena race conditions terjadi ketika beberapa thread bersaing untuk mengakses dan memodifikasi data yang sama secara bersamaan, tanpa ada mekanisme sinkronisasi yang tepat. Sementara dalam kasus ini, fungsi `threaded_unbounded_lap2d!` bekerja dengan prinsip setiap thread bekerja pada area yang berbeda dalam matriks u dan unew.

Pada setiap iterasi loop, setiap thread bekerja pada indeks baris i yang berbeda. Misalnya, thread 1 bekerja pada i = 2, thread 2 pada i = 3, dan seterusnya. Oleh karena itu, mereka tidak saling bersaing untuk mengakses dan memodifikasi elemen yang sama secara bersamaan.

Selain itu, tidak ada penugasan nilai yang bergantung pada hasil perhitungan sebelumnya dalam loop. Setiap elemen matriks unew hanya bergantung pada elemen-elemen yang sesuai dalam matriks u. Oleh karena itu, tidak ada ketergantungan antara iterasi yang berpotensi menyebabkan race conditions.

Semua operasi di atas dijalankan dengan jumlah threads terpakai sebanyak 2 threads.

2

```
1 nprocs()
```

Only process 1 can add and remove workers

1. **cluster_mgmt_from_master_check** @ *cluster.jl:993* [inlined]
2. **var"#addprocs#41"(::Base.Pairs{Symbol, Union{}, Tuple{}, NamedTuple{(), Tuple{}}}, ::typeof(addprocs), ::Distributed.LocalManager)** @ *cluster.jl:446*
3. **addprocs** @ *cluster.jl:443* [inlined]
4. **#addprocs#262** @ *managers.jl:465* [inlined]
5. **addprocs** @ *managers.jl:462* [inlined]
6. **top-level scope** @ *Local: 1* [inlined]

```
1 addprocs(1)
```

Saat ingin menambah worker process, sayangnya tidak bisa dilakukan. Mungkin karena keterbatasan perangkat yang digunakan.