PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
INSTITUTO DE INGENIERÍA MATEMÁTICA Y COMPUTACIONAL

**IMT2112**
**2do semestre del 2021**

**Nombre: Agustín Caracci**

# Tarea 2

## 1.

El código se encuentra en el archivo `sequential.cpp`. El área calculada es de 1,504288, lo cual es bastante cercano al valor esperado. Los números aleatorios fueron generados usando la librería `<random>`. El tiempo que tardó el código en ejecutarse fue 29.079 segundos.

## 2.

El código se encuentra en el archivo `parallel.cpp`. Se probó usando $2, 4, 8$ y $16$ *threads*. El tiempo que tomó con cada cantidad fue 13.506, 9.869, 9.961 y 9.849, respectivamente. Se redujo el tiempo en casi dos tercios.

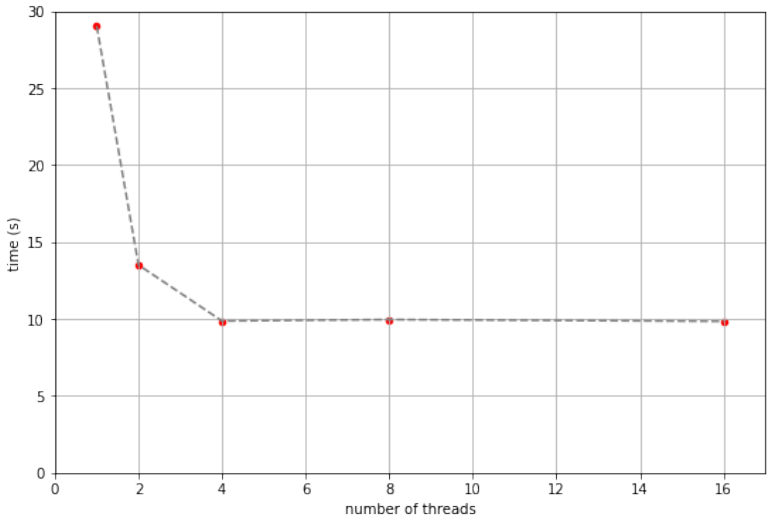


Figura 1: Resultados

## 3.

El método se sostiene en el teorema del límite central, ya que la probabilidad de que un número al azar en un cuadrado que contiene al conjunto, pertenezca al conjunto, es la proporción del área del conjunto con respecto al área del cuadrado. Luego, mientras más números aleatorios se generan, la proporción de números que pertenece al conjunto con respecto a los números totales va a tender a la proporción del área del conjunto.

Para todos los experimentos se usaron 500,000 números complejos aleatorios en el intervalo $[-2, 2]$ tanto para la parte real como la parte imaginaria. Para decidir si un número pertenecía al conjunto o no, se verificó que $|x_n| \leq 2$ para la sucesión $x_0 = 0,\ x_n = x_{n-1}^2 + c,\ n = 1, \ldots, 2000$. Si este era el caso, se consideraba que el número pertenecía al conjunto. Luego, se contaban cuantos números pasaban el criterio recién descrito, se dividía esa cantidad por la cantidad total de números aleatorios generados (500.000) y se multiplicaba esto por el área del espacio muestral (16). El resultado obtenido es la aproximación del conjunto de Mandelbrot.

(a) No se puede paralelizar porque la generación de números seudo-aleatorios es un proceso secuencial determinista, donde la generación de cada número depende del número anterior. Luego, si se paralelizara este proceso en $n$ *threads*, se obtendrían $n$ secuencias aleatorias iguales, pues cada thread empezaría con el mismo número inicial y a partir de este los números se generan de manera determinista. Sin embargo, existen maneras de evitar estos problemas, como que cada hilo inicie con una semilla distinta, lo cual se puede realizar de distintas maneras (por ejemplo, la semilla puede ser el id del *thread*). Existen librerías de c++ que permiten la generación de números aleatorios en paralelo, como la librería `<random>`. Una discusión mas avanzada en el tema se puede encontrar aquí.

(b) Se creó un *array* del tamaño de la cantidad de *threads*, donde cada entrada estaba destinada para ser el contador de cada *thread* individual. Luego, no es posible que haya corrupción de los datos por parte de distintos hilos por falta de comunicación. Cuando se terminaban todos los hilos, se sumaban las sumas parciales guardadas en el *array* para obtener el valor buscado.