Implementación y análisis de los algoritmos de Monte Carlo Needles y Dartboard aplicado al cálculo de PI.

Presentado por:

Jhonattan Alcaraz Carvajal
Diego Fernando Gómez
David Andrés Patiño

Presentado a:
Ramiro Andrés Barrios Valencia

HPC: High Performance Computing
Ingeniería de Sistemas y Computación
Universidad Tecnológica de Pereira
2022

Contenido

Resumen

En este documento se muestra el proceso realizado para llevar al lenguaje de programación C++ el algoritmo de MonteCarlo Needles y Dartboard para el cálculo de PI. Este algoritmo usa como bases fundamentales la estadística y la probabilidad, ya que está basado en casos de éxito entre múltiples intentos, y permite obtener más valores y resolución para la parte decimal de la constante en la medida que el número de intentos sea mayor. La ejecución de este algoritmo se realizó primeramente de forma secuencial. Con el fin de reducir el tiempo de espera para la obtención de resultados se han implementado hilos y procesos. Por último, se realiza un análisis a los resultados obtenidos.

Introducción

Uno de los descubrimientos más importantes en la historia de la matemática fue la constante PI. Este número irracional, que relaciona la longitud de una circunferencia y su diámetro en la geometría euclidiana, tienen una característica especial: tener cifras infinitas en su parte decimal.

El nombre actual del número, el mismo que el de la letra griega inicial de las palabras 'periferia' y 'perímetro', fue usado primero por William Oughtred (1574-1660), aunque lo popularizó Leonhard Euler (1707-1783). Antes, el número pi había sido conocido como 'constante de Ludolph' (en honor al matemático Ludolph Van Ceulen) o como 'constante de Arquímedes'.

En el siglo III antes de Cristo, el físico griego utilizó polígonos para afinar el cálculo y llegó a determinar el valor de pi con un error de entre 0.024% y 0.040% sobre el valor real.

Siguió con la misma práctica Claudio Ptolomeo (en el siglo II de nuestra era), quien mejoró la aproximación de Arquímedes, y estableció el valor de 3,14166 para PI empleando un polígono de 120 lados. A finales del siglo V, el matemático y astrónomo chino Zu Chongzhi dio un paso más, atribuyéndole un valor de 3.1415927, resultado que no fue mejorado hasta el siglo XV.

Tras siglos de cálculos, la llegada de los ordenadores cambió los parámetros: el cálculo se disparó y se han ido añadiendo decimales a π hasta los 12,1 billones actuales.

Usando la tecnología que se dispone en la actualidad se usan dos algoritmos, el Buffon Needle y Dartboard, que pueden ser implementados en el lenguaje de programación C++.

Marco conceptual

High Performance Computing

Cuando hablamos de HPC por sus siglas en inglés, o Computación de alto rendimiento en español, “hacemos referencia a un campo de la computación actual que da solución a problemas tecnológicos muy complejos y que involucran un gran volumen de cálculos o de coste computacional.” (López, 2017) Para nuestro caso, al aumentar la dimensión de las matrices a multiplicar, esto supone un alto costo computacional, el cual, según el análisis, se ve disminuido gracias al uso de herramientas que permiten, precisamente, disminuir el tiempo de ejecución.

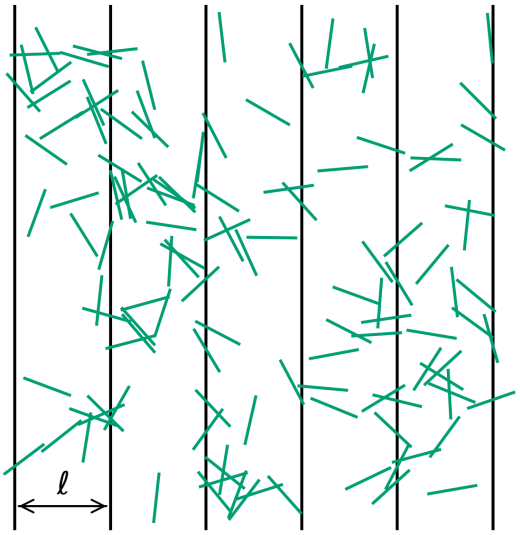
Algoritmo de MonteCarlo

El método de Montecarlo es un método no determinista o estadístico numérico, usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. El método se llamó así en referencia al Casino de Montecarlo (Mónaco) por ser “la capital del juego de azar”, al ser la ruleta un generador simple de números aleatorios. El nombre y el desarrollo sistemático de los métodos de Montecarlo datan aproximadamente de 1944 y se mejoraron enormemente con el desarrollo de la computadora.

Buffon’s Needles

Es un problema clásico de probabilidad geométrica que consiste en lanzar agujas sobre un papel en el que se han trazado líneas rectas paralelas distanciadas entre sí de manera uniforme. Se puede demostrar que si la distancia entre las rectas es igual a la longitud de la aguja, la probabilidad de que la aguja cruce alguna de las líneas es 2/π.
Tal que: $\pi = (2*N)/A$, donde N es el número total de intentos y A el número de veces que la aguja ha cruzado alguna línea

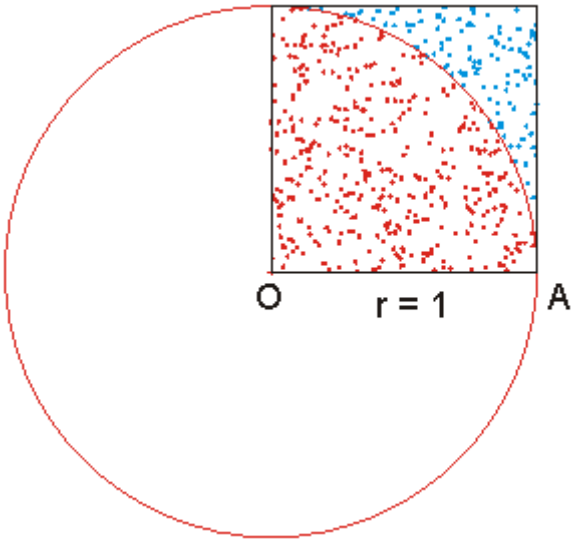
Este método se puede visualizar fácilmente en la siguiente imagen.



El algoritmo Dartboard

Para este problema se toma un círculo con radio r=1 delimitado por un cuadrado de tamaño 2*r=2. Se lanzan dardos al azar dentro del cuadrado siguiendo una distribución aleatoria, se cuenta como acierto cada vez que un dardo golpea dentro del círculo. Para facilitar el cálculo se divide la figura en cuadrantes con un plano cartesiano, tomando así el cuadrante donde los ejes son positivos, si lanzamos los dardos a este cuadrante tendremos que: $\pi = (4*A)/N$, donde N es el número total de lanzamientos en el cuadrante y A el número de veces que el dardo acierta en el interior del círculo.

Este método se puede visualizar fácilmente en la siguiente imagen.



Complejidad Computacional

“La Teoría de la Complejidad estudia la eficiencia de los algoritmos en función de los recursos que utiliza en un sistema computacional (usualmente espacio y tiempo)” (Cortéz, 2004). Para este estudio, la complejidad computacional permite observar que en la multiplicación de matrices se gastan bastantes recursos de procesamiento en cuanto a tiempo, es decir, para matrices de dimensiones muy grandes, el procesador estará ocupado bastante tiempo resolviendo la multiplicación, lo que motiva aún más a programar de manera paralela el algoritmo, aprovechando al máximo los recursos del procesador.

Programación paralela

“La computación paralela es el uso de múltiples recursos computacionales para resolver un problema. Se distingue de la computación secuencial en que varias operaciones pueden ocurrir simultáneamente.” (Ferestrepoca, 2019) En nuestro caso de estudio se ejecutará el código de multiplicación de matrices de forma secuencial y de forma paralela, esto con el fin de comparar los resultados obtenidos por cada ejecución y de esta manera poder analizar la eficiencia de la programación paralela.

Hilos

“La diferencia fundamental entre hilos y procesos es que un proceso dispone de un espacio de memoria separado, mientras que un hilo no. De esta forma los hilos pueden trabajar directamente con las variables creadas por el proceso padre, mientras que los procesos hijos no pueden hacerlo.” (Montero & Antunez, 2011) Esta información nos facilita la comprensión del principio aplicado para esta implementación.

Speedup

El speedup representa la ganancia que se obtiene en la versión paralela de un programa con respecto a su versión secuencial. Para este estudio se ha tomado el speedup como:

$$Speedup = \frac{T*(n)}{T_P(N)}$$

Donde T*(n) hace referencia al tiempo de ejecución secuencial y Tp (N) hace referencia al tiempo de ejecución paralelo

Marco Contextual

Los algoritmos Monte Carlo Needles y Dartboard para el cálculo de PI se probaron con lanzamientos de 10, 100, 1000, 10000, 100000 y 1000000. El valor de PI que se usará como el “real” es 3.14159265. Se analizarán los distintos métodos evaluando los recursos consumidos como el tiempo; se sacaron algunas gráficas para mostrar los resultados obtenidos.

Características de la máquina

- Procesador: Intel(R) Core(TM) i7-10700
- Cantidad de núcleos: 8
- Cantidad de subprocesos/hilos: 16
- Frecuencia básica del procesador: 2.90GHz
- Frecuencia Turbo máxima: 4.80GHz
- Ram 16GB DDR4 @ 2400 MHz
- SSD: 240GB - R: 540 MB/s - W: 500 MB/s
- Sistema operativo: Ubuntu 20.04.4 LTS

Desarrollo

Para que el algoritmo funcione correctamente en la máquina se pueden usar técnicas que permitan al equipo, procesar de la mejor forma posible la información buscando reducir el tiempo necesario para dar el resultado que contenga el valor experimental de pi, o bien usar una versión secuencial, con mas tiempo de ejecucion, pero menor complejidad de implementación.

Para complementar el análisis de los datos y tener una vista un poco más clara de la información obtenida, se han utilizado las tablas para hacer gráficas comparativas que nos permitan visualizar la eficiencia de trabajar con hilos.

Ejecución secuencial

A continuación se presentarán los datos obtenidos al ejecutar el código de forma secuencial de los algoritmos de Needles y Dartboard.

- Dartboard

Tabla 1: Resultados de la ejecución secuencial dartboard.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error

10	0.0004	4	27.32%
100	0.0023	3.24	3.13%
1000	0.0238	3.112	0.94%
10000	0.1928	3.1356	0.19%
100000	1.5794	3.15184	0.33%
1000000	12.619	3.14291	0.04%

- Needles

Tabla 2: Resultados de la ejecución secuencial needles.

Lanzamientos	Tiempo	Valor de PI obtenido	Porcentaje de Error
10	0.013	3.3333	6.10%
100	0.0163	3.0303	3.54%
1000	0.0542	3.19489	1.70%
10000	0.396	3.15457	0.41%
100000	2.9537	3.13548	0.19%
1000000	30.8277	3.14354	0.06%

Ejecución con dos hilos

A continuación se presentarán los datos obtenidos al ejecutar el código de los algoritmos de Needles y Dartboard usando dos hilos. A partir de estos datos aparece la fila de speedup con el fin de observar la ganancia obtenida usando paralelización.

- Dartboard

Tabla 3: Resultados de la ejecución con dos hilos dartboard.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error	SpeedUp
10	0.1267	2.8	10.87%	0.0032
100	0.1121	3.08	1.96%	0.0205
1000	0.1066	3.136	0.18%	0.2233
10000	0.1598	3.1624	0.66%	1.2065
100000	0.3933	3.1474	0.18%	4.0158
1000000	3.2841	3.1408	0.03%	3.8425

- Needles

Tabla 4: Resultados de la ejecución con dos hilos needles.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error	SpeedUp
10	0.1328	1.66667	46.95%	0.0979
100	0.1121	3.0303	3.54%	0.1454
1000	0.1235	2.99401	4.70%	0.4389
10000	0.1118	3.05904	2.63%	3.5420
100000	0.373	3.14842	0.22%	7.9188
1000000	2.7232	3.14739	0.18%	11.3204

Ejecución con cuatro hilos

A continuación se muestran los valores que resultan de la ejecución de los algoritmos Needles y Dartdoard con cuatro hilos

- Dartboard

Tabla 5: Resultados de la ejecución con cuatro hilos dartboard.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error	SpeedUp
10	0.1626	3.2	1.86%	0.0025
100	0.1099	3.16	0.59%	0.0209
1000	0.1054	3.156	0.46%	0.2258
10000	0.1119	3.1372	0.14%	1.723
100000	0.2802	3.14596	0.14%	5.6367
1.00E+06	2.0442	3.14027	0.04%	6.1731

- Needles

Tabla 6: Resultados de la ejecución con cuatro hilos needles.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error	SpeedUp
10	0.1942	5	59.15%	0.0021
100	0.1839	2.77778	11.58%	0.0125
1000	0.1713	2.94118	6.38%	0.1389
10000	0.1675	3.10078	1.30%	1.151
100000	0.2501	3.16766	0.83%	6.3151
1.00E+06	1.5056	3.13593	0.18%	8.3814

Ejecución con ocho hilos

A continuación se muestran los resultados de la ejecución con 8 hilos para ambos algoritmos

- Dartboard

Tabla 7: Resultados de la ejecución con ocho hilos dartboard.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error	SpeedUp
10	0.3399	2	36.34%	0.0012
100	0.2815	2.84	9.60%	0.0082
1000	0.2593	3.08	1.96%	0.0918
10000	0.2586	3.1392	0.08%	0.7456
100000	0.3355	3.13696	0.15%	4.7076
1.00E+06	2.3244	3.14112	0.02%	5.4289

- Needles

Tabla 8: Resultados de la ejecución con ocho hilos needles.

Lanzamientos	Tiempo(ms)	Valor de PI obtenido	Porcentaje de Error	SpeedUp
10	0.3505	2.5	20.42%	0.0011
100	0.2483	4.7619	51.58%	0.0093
1000	0.1933	3.03951	3.25%	0.1231
10000	0.2597	3.15259	0.35%	0.7424
100000	0.3011	3.1498	0.26%	5.2454
1.00E+06	1.5294	3.13982	0.06%	8.2509

Ejecución con optimización O2

El compilador gcc tiene opciones de optimización implementadas internamente, de las cuales se utilizara la opción O2 de este compilador con el código secuencial de ambos algoritmos, además se mostrará el speed up respectivo de las dos ejecuciones.

- Dartboard

Tabla 9: Resultados de la ejecución con optimización O2 dartboard.

Lanzamientos	tiempo(ms)	valores de pi	porcentaje de error	Speedup
1000	0.027	3.12280	0.5982%	0.014814815
10000	0.232	3.14724	0.1798%	0.009913793
100000	2.957	3.14151	0.0026%	0.008048698
1000000	25.076	3.14224	0.0206%	0.007688627
10000000	194.809	3.14155	0.0014%	0.008107428
100000000	1,837.229	3.14159	0.0001%	0.006868496

- Needles

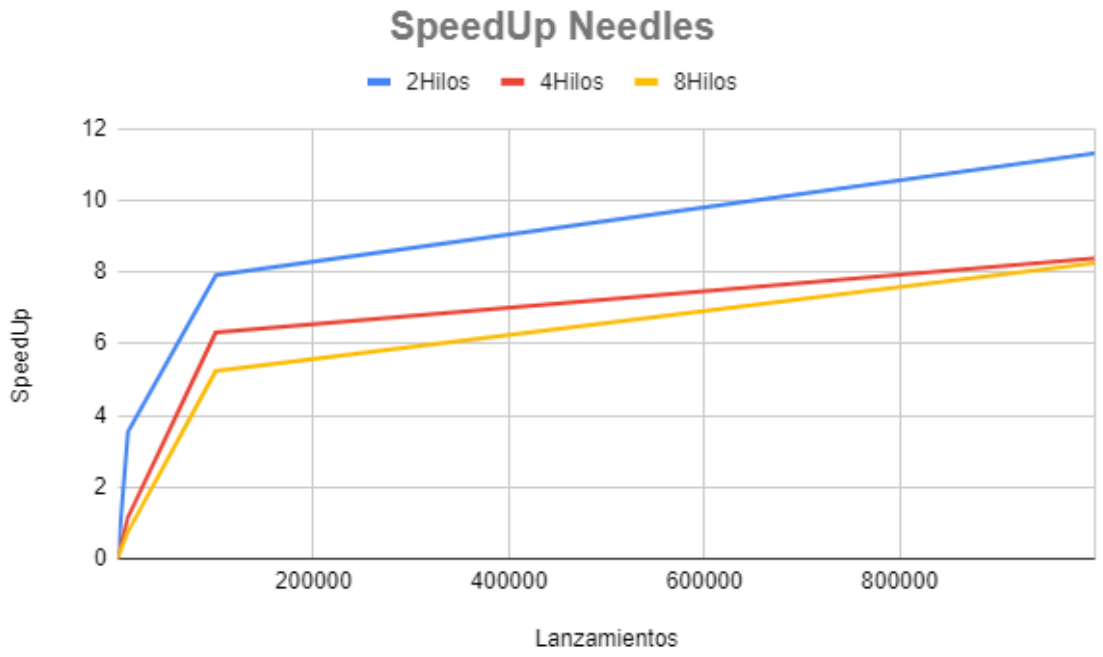
Tabla 10: Resultados de la ejecución con optimización O2 needles.

Lanzamientos	tiempo(ms)	valores de pi	porcentaje de error	Speedup
1000	0.071	3.2044	1.9983%	0.00563380
10000	0.511	3.1524	0.3431%	0.00450098
100000	4.681	3.1396	0.0634%	0.00508438
1000000	47.697	3.1405	0.0348%	0.00404218
10000000	378.864	3.1403	0.0418%	0.00416878
100000000	3,740.094	3.1416	0.0015%	0.00337398

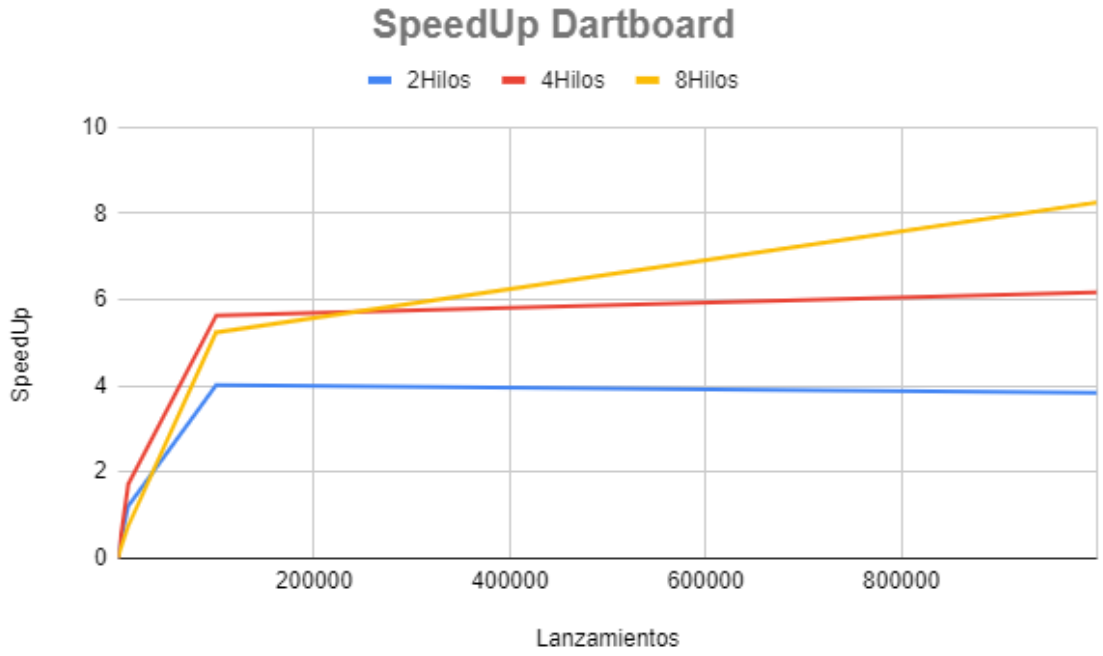
Speed Up

En la siguiente gráfica se muestra una comparativa entre todos los speedUp obtenidos en cada ejecución del código donde se hacía uso de paralelización mediante hilos, ya sean 2, 4 u 8 hilos.

Gráfica 1: Comparación entre SpeedUp de la ejecución del algoritmo Needles con 2,4 y 8 hilos.

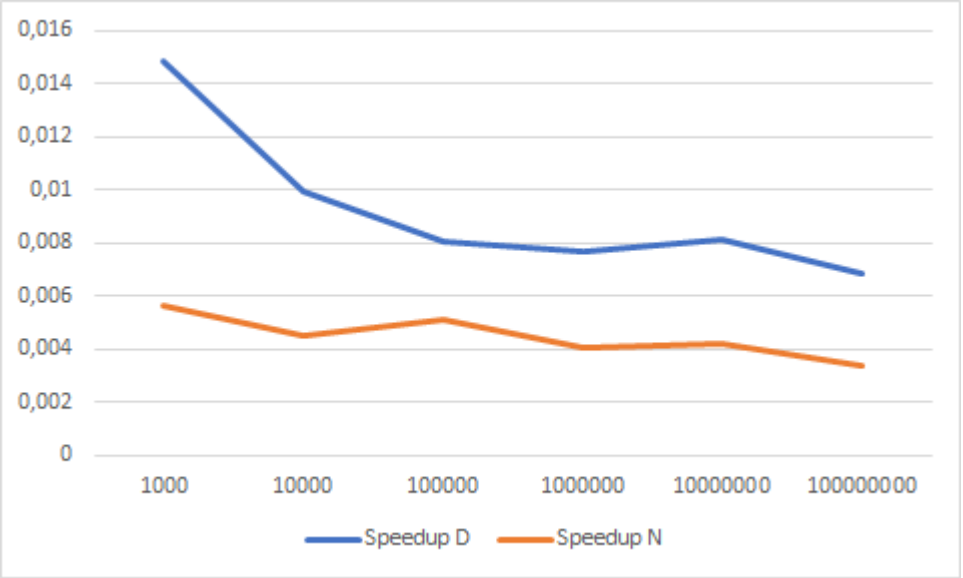


Gráfica 2: Comparación entre SpeedUp de la ejecución del algoritmo DartBoard con 2,4 y 8 hilos.



A continuación se presentan los resultados del speed up obtenido en las ejecuciones con la optimización O2 en ambos algoritmos

Gráfica 3: Comparación entre SpeedUp de la ejecución de cada algoritmo con optimización O2.



Conclusiones

- Al llevar a cabo la ejecución de los algoritmos, se pudo observar que las aproximaciones del número pi que arrojaban las pruebas fueron muy acertadas, puesto que siempre se aproximaban al valor ya conocido. Lo cual permite concluir que los algoritmos Needle y Dartboard funcionan, demostrando ser un gran aporte a la ciencia.

- Al observar las gráficas del SpeedUp es posible observar que los algoritmos presentan una mejora bastante notable al ser implementados con hilos, esto se debe a que cada hilo se encarga de una tarea liviana en comparación con el número de tareas que tiene que realizar un mismo proceso en el caso de la ejecución secuencial.

- Usar más de ocho hilos es un desperdicio de recursos debido a que no hay un aumento considerable en el speedup.

- En el algoritmo de Needles es muy extraño que el SpeedUp de dos hilos sea superior al de ocho hilos. Esto puede deberse a problemas en la toma de los datos.

Bibliografía

- Arndt Jörg, Haenel Christoph. (2001). Pi - Unleashed. Springer Science & Business Media. Recuperado de: https://books.google.com.co/books?id=JlG5rFH7Ge0C&pg=PA39&lpg=PA39&dq=Dartboard+Method+algorithm&source=bl&ots=t76R30Q342&sig=NjguOYMc0ILqZs8Bcz6uIpfejdc&hl=en&ei=-YzTSuutFMefkQXj9_H7Aw&sa=X&oi=book_result&ct=result&redir_esc=y#v=onepage&q&f=false
- Arthur York, Cory Simon. (2018). Monte Carlo simulation of Buffon 's Needle. The Simon Ensemble. Recuperado de: https://simonensemble.github.io/2018-04/buffon
- (2018). Día de Pi: ¿Quién descubrió el valor del número pi?. ElPeriódico. Recuperado de: https://www.elperiodico.com/es/extra/20180314/dia-de-pi-2018-6687376
- (2008). Método de Montecarlo. Wikipedia. Recuperado de: https://es.wikipedia.org/wiki/Método_de_Montecarlo
- Mulkers Jeroen. (2013). BuffonNeedle. GitHub. Recuperado: https://github.com/JeroenMulkers/BuffonNeedle