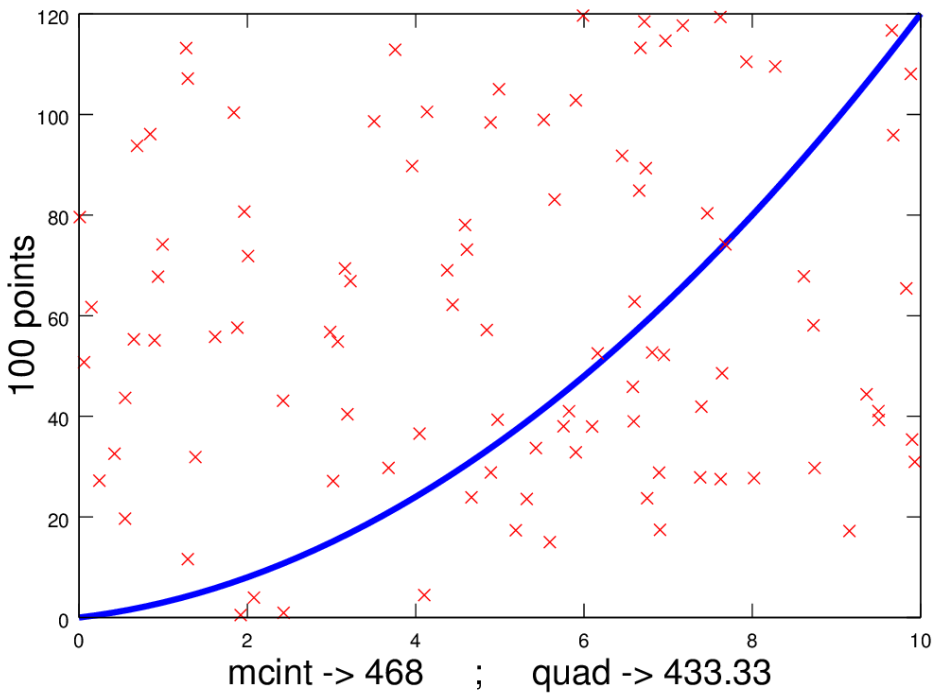
Práctica 0: Octave

Esta práctica consiste en implementar una función que calcule la integral por el método de Monte Carlo.

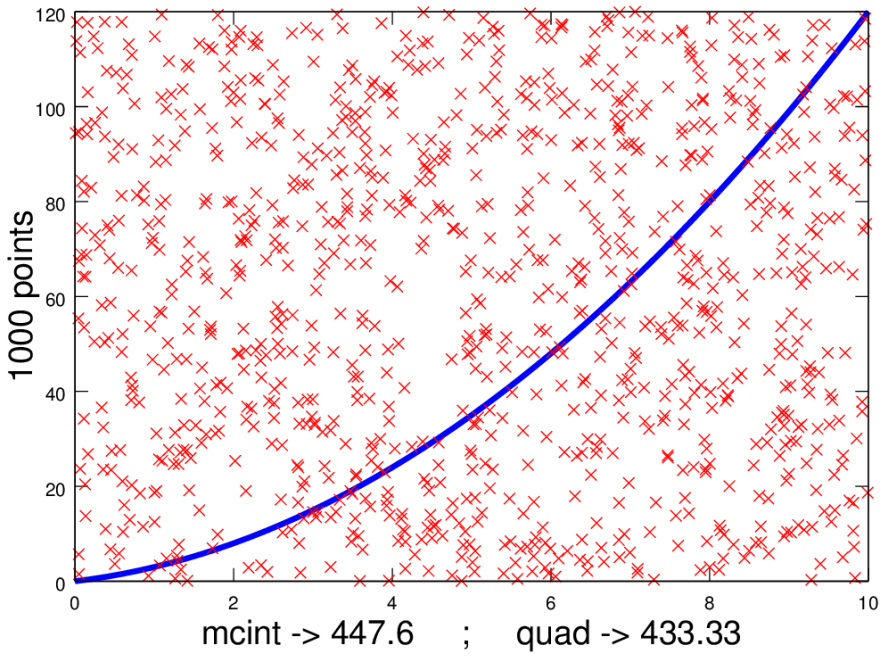
Este método se basa en generar n puntos aleatorios, y calcular cuántos de ellos se sitúan por debajo de la curva de la función en cuestión. Es un método intuitivo basado en la definición geométrica de la integral: “el área limitada por la gráfica de la función”.

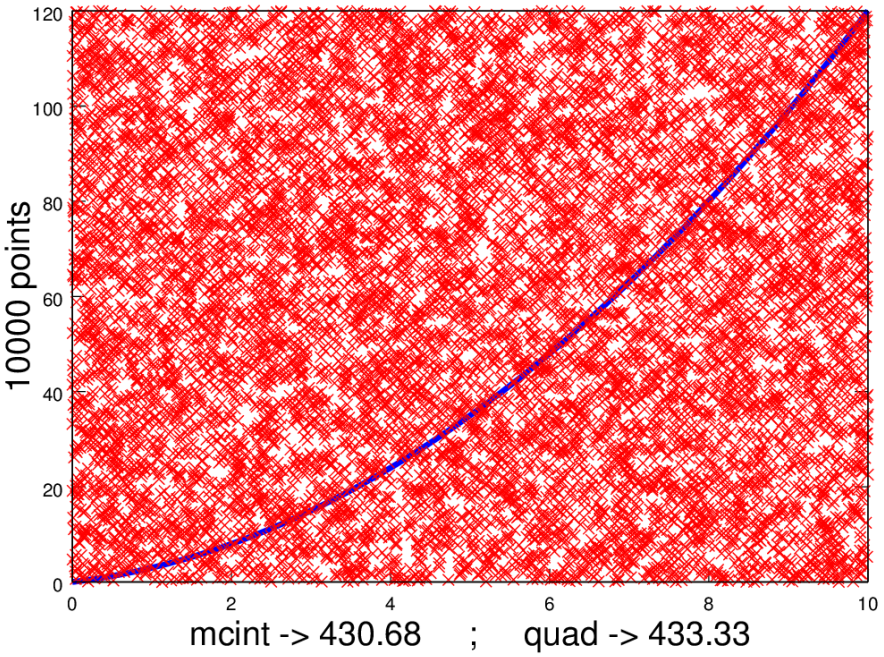
Como no es un método matemático, los resultados que se obtienen son valores aproximados, y dependen además del número de puntos aleatorios que se generen. El resultado será más preciso cuantos más números se generen.

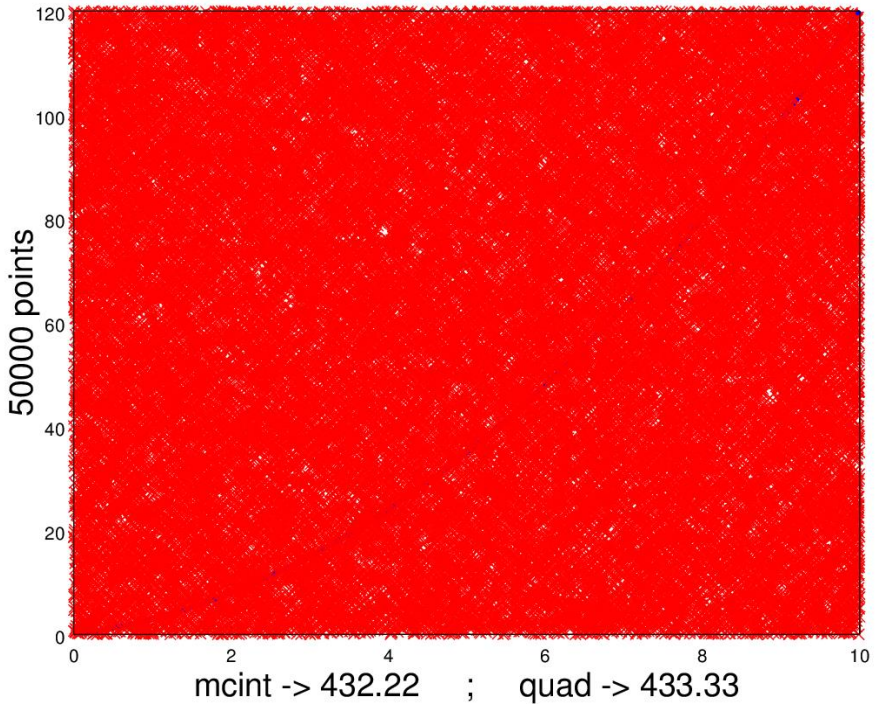
Como muestra de ello, podemos ver las siguientes gráficas, en la que se compara el resultado con el método de Monte Carlo y con la función “quad” que ofrece Octave. En este ejemplo se emplean 100, 1000, 10000, y 50000 puntos.



Apr r


1000 points
mcint -> 447.6 ; quad -> 433.33


10000 points
mcint -> 430.68 ; quad -> 433.33


50000 points
mcint -> 432.22 ; quad -> 433.33

Comparación de tiempo de los distintos algoritmos

Al implementar dicha función en Octave, al generar los puntos aleatorios se pueden emplear 2 métodos:

- Empleando operaciones entre vectores
- Empleando bucles

La diferencia entre ambos métodos se encuentra en el rendimiento del cálculo, ya que Octave está optimizado para emplear operaciones entre vectores, y es notablemente más rápido en este primer método.

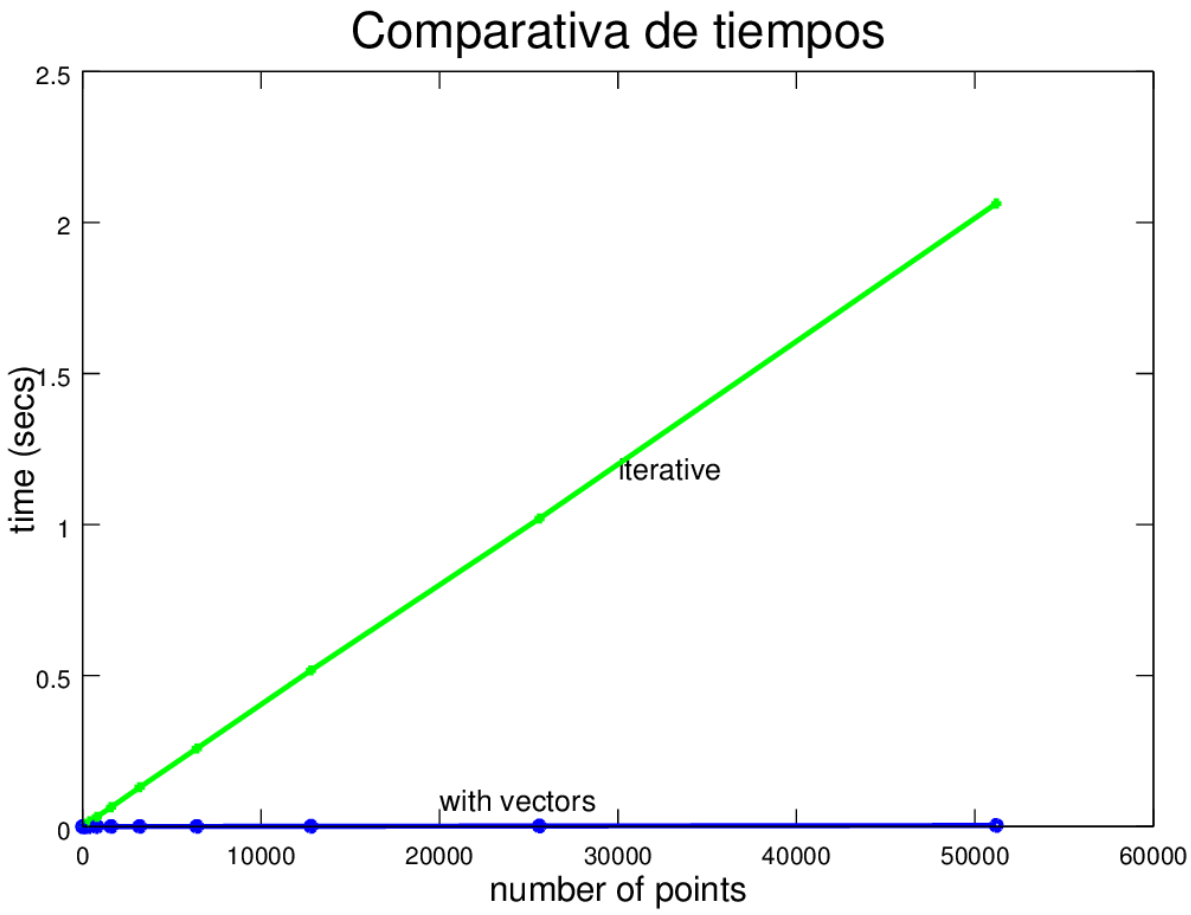
Para realizar una comparativa de estos 2 métodos se ha realizado un test en el que se ejecutan estas 2 funciones para distintas cantidades de puntos aleatorios, y después se comparan ambos tiempos.

Para que los tiempos de ejecución sean más precisos, e influyan lo menos posible otros factores externos a los propios algoritmos, se ha decidido realizar el test repetidas veces (de 20 a 200 veces, dependiendo del número de puntos generados) y luego se ha calculado el tiempo medio.

Los resultados han sido los siguientes:

Tabla de tiempos

Nº de puntos	100	200	400	800	1600	3200	6400	12800	25600	51200
mcint (vect)	1,30E-04	1,30E-04	1,36E-04	1,80E-04	2,25E-04	3,33E-04	5,36E-04	1,00E-03	2,04E-03	3,85E-03
mcint (itera)	0,00409	0,00803	0,01597	0,0322	0,064	0,12883	0,25609	0,51119	1,0263	2,0434



Como se puede comprobar, el coste del algoritmo iterativo es notablemente superior, siendo este prácticamente lineal en función del número de puntos, ya que al duplicar el número de puntos de duplica el tiempo de ejecución.

En cambio, el algoritmo con vectores tiene un coste casi constante ya que va creciendo en función del número de puntos, pero su crecimiento es prácticamente inapreciable.

En las siguientes páginas se puede ver el código fuente de los algoritmos utilizados.