Informe de Laboratorio N°3

Alumno: Méndez Martín
Docentes: Dra. Marconi Verónica I.; Dr. Banchio Adolfo
Universidad Nacional de Córdoba (UNC)
Facultad de Matemática, Astronomía, Física y Computación (FaMAF)
Curso de Física Computacional: Problema N°4

I. Introducción

I-1. Problema 4 - La Hiperesféra

Calcular numéricamente el volumen $V(n)$ de una esfera n-dimensional de radio 1, integrando la función $f(x_1,\ldots,x_{(n-1)}) = 2\cdot\sqrt{1-\sum_{i=1}^{(n-1)}(x_i)^2}$, sobre la esfera $(n-1)$-dimensional de los siguientes modos:

- a) Utilizando, para $n = 2, 3, 4$, el método del trapecio, generalizado a n dimensiones. Elija una cantidad de puntos, $N_p = 2^{24}$, y manteniéndolo fijo, grafique el error relativo de la integral versus n. Encuentre la dependencia esperada.
- b) Utilizando el método de Monte Carlo. Éste debe funcionar hasta al menos $n = 100$ (dando el valor correcto al 1 por mil de error en unos pocos minutos de CPU). Observe que el volumen de una hiperesféra rápidamente se hace mucho menor que el del hipercubo (una distribución uniforme muestrearía el hipercubo, por lo tanto, arrojaría casi todos los puntos fuera del dominio de integración). Además tenga en cuenta que para la distribución Gaussiana unidimensional $\langle x^2\rangle = \sigma^2$.

Nota: el resultado analítico para n par es $V(n) = \frac{\pi^{n/2}}{(n/2)!}$ y para n impar es $V(n) = \frac{\pi^{(n-1)/2}\cdot 2^n\cdot[(n-1)/2]!}{n!}$. Para $n = 100$ da $V(100) = 2{,}3682\cdot\cdots\cdot 10^{-40}$

I-2. Método del trapecio

Para computar el cálculo del volumen de la hiper-esféra con el método del trapecio se utilizó la fórmula recursiva que nos permite relacionar el volumen de una esféra n-dimensional con una esféra $(n-1)$-dimensional, a través de una integración de la forma

$$V_n(R) = V_{n-1}(R)\int_{-R}^{R}\left[1-\left(\frac{x}{R}\right)^2\right]^{(n-1)/2} dx \qquad (1)$$

En nuestro caso particular, el radio $R = 1$ y cómo se trata de una geometría completamente simétrica sólo integramos desde 0 hasta 1, con la salvedad de multiplicar el volumen n-dimensional obtenido por un factor de 2^n.

II. Resultados y Discusiones

II-A. Método del trapecio

Teniendo en cuenta que se debe mantener fijo el número de puntos ($N_p^{nD} = 2^{24}$) utilizados para la integración de un volumen n-dimensional. A medida que se aumenta la dimensión cada integral unidimensional debe utilizar ($N_p^{1D} = (N_p^{nD})^{1/n}$) de esta forma logramos recuperar la dependencia correcta del error relativo $\epsilon_{rel} = \frac{\epsilon_{exact}-\epsilon_{aprox}}{\epsilon_{exact}}$ con el número de dimensiones. Las simulaciones con este método calcularon para $n = 1, 2, 3, 4$ y los resultados obtenidos fueron,

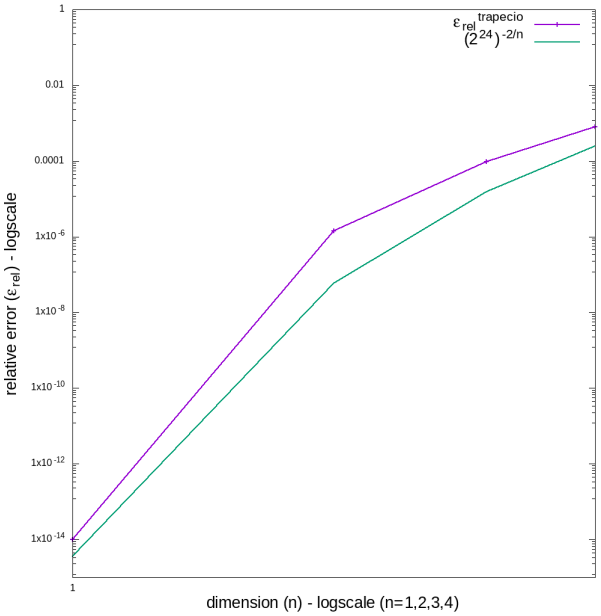

Fig. 1. error relativo (regla trapecio) vs dimensiones

Notemos que el *fitteo* no es exacto, lo cual podría deberse a algún error en el algoritmo. Sin embargo, las pendientes tienen bastante coincidencia. Este resultado, nos permite concluir que, como el método de Monte Carlo es independiente del número de dimensiones para dimensiones mayores a 4 el método de Monte Carlo saca ventaja sustancial con respecto al método del trapecio y para dimensiones mayores a 6 respecto al método de Simpson.

II-A1. Determinación de la desviación estandar σ

Teniendo en cuenta que la razón entre los volumenes n-dimensionales de la hiper-esféra y el del hiper-cubo disminuye a medida que el número de dimensiones n aumenta, y recordando que una distribución uniforme muestrearía completamente el hiper-cubo, se utilizó el método de *importance sampling*, con distribución gaussiana, para ser más eficiente con el método de MC, es decir, lograr que la mayor cantidad de puntos aleatorios caigan dentro del volumen a calcular (volumen de la hiper-esféra), sin embargo, a

medida que aumentamos las dimensiones de la hiper-esfera se debe modificar la desviación estándar de la distribución para lograr una precisión fija admisible. Lo primero que se hizo fue correr el programa para una cantidad de números aleatorios fija $n_{random} = 10^6$, definir el rango de variación de la desviación estándar ($\sigma_{máx} = 1; \sigma_{mín} = 0{,}01$) y el numero de valores para σ distintos que quiero que se calcule $n_\sigma = 1000$. Con estos valores se define un paso $\Delta\sigma$ y se comienza la simulación partiendo con un $\sigma = \sigma_{mín}$, al obtener el resultado del volumen se computa el error relativo $\epsilon_{rel} = \frac{\epsilon_{exact} - \epsilon_{aprox}}{\epsilon_{exact}}$ y nos preguntamos si este error es menor a 0,001 (correcto en uno por mil) en caso de ser afirmativo se termina la simulación y en caso de ser negativo se aumenta en uno el paso ($\sigma = \sigma + \Delta\sigma$) y se vuelve a computar el volumen. Estas corridas se hacen para un numero de dimensiones reducido, es decir, $1 \leq n \leq 7$. Los resultados obtenidos se resumen en la siguiente tabla I.

CUADRO I
PRIMER TESTEO DE σ

Dimensión	Elapsed time [s]	ϵ_{rel}	σ
1	0.2470E+02	0.3298E-03	0.2250
2	0.4426E+02	0.4368E-03	0.2141
3	0.5253E+02	0.8824E-03	0.1686
4	0.5659E+02	0.1947E-03	0.1408
5	0.1130E+03	0.8072E-03	0.2181
6	0.1105E+03	0.4670E-04	0.1765
7	0.1412E+03	0.4518E-03	0.1953

Notar que también se computó el tiempo de CPU transcurrido, es decir, el tiempo en que tarda el algoritmo en encontrar un $\sigma_{óptimo}$ tal que verifique el error propuesto, estos tiempos van incrementando con el número de dimensiones. Además, notemos que el $\sigma_{óptimo}$ para dimensiones mayores a 1 siempre es menor al $\sigma_{óptimo}$ de una dimensión, pero de forma general $\sigma_{óptimo}$ no disminuye con el numero de dimensiones (este tema se trata más adelante). Estos dos resultados, nos permiten asegurarnos que el mayor valor de σ se obtiene para $n = 1$, lo cual nos permitiría reducir significativamente los tiempos de CPU acotando el rango de variación de la desviación estándar, es decir, para más dimensiones consideraremos ($\sigma_{máx} = 1; \sigma_{mín} = 0{,}1$).

Por otro lado, como nosotros sabemos que a medida que aumenta el número de dimensiones la razón entre el volumen del la hiper-esféra y el del hiper-cubo se reduce, la distribución gaussiana debe disminuir su desviación estándar para mejorar la precisión del método y lograr que la mayor cantidad de números computados caigan dentro del volumen a calcular, el valor $\sigma = 0$ será una especie de *atractor* y a medida que aumenta el número de dimensiones el valor de $\sigma_{óptimo}$ tratará de desviarse lo menos posible del *atractor* tendiendo, en el límite de $n \to \infty$, a cero y la distribución gaussiana tenderá a una delta de Dirac. Es por ello que, para reducir significativamente los tiempos de CPU se propuso una "ley"(ad-hoc) de variación de $\Delta\sigma$ con el número n de dimensiones de la siguiente manera

$$n \in N \Rightarrow n_\sigma = 100 \cdot n \Rightarrow \Delta\sigma = \frac{\sigma_{máx} - \sigma_{mín}}{(n_\sigma - 1)}$$

$$\Rightarrow \Delta\sigma = \frac{\sigma_{máx} - \sigma_{mín}}{(100 \cdot n - 1)} \quad (2)$$

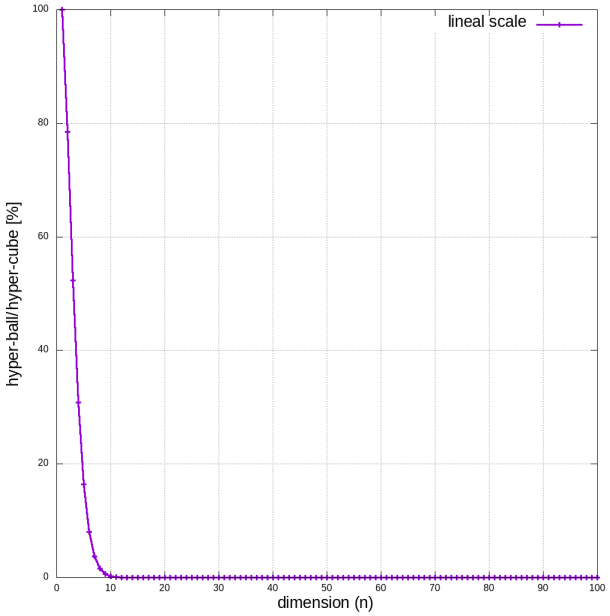

Fig. 2. razon entre volumenes (ball/cube) vs dimensiones

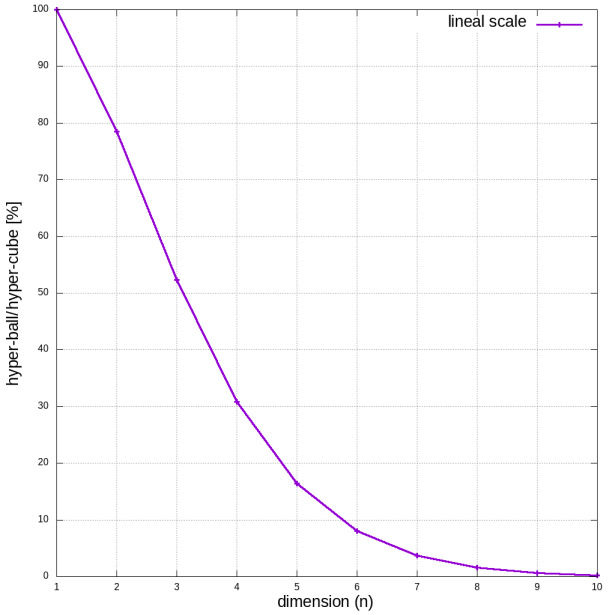

Fig. 3. razon entre volumenes (ball/cube) vs dimensiones

De esta forma, a medida que aumenta el número de dimensiones se computa muy cerca del valor mínimo propuesto para la desviación estándar y se itera hasta conseguir el error admisible. Con esta propuesta se calcularon los volúmenes de la hiper-esféra hasta 10 dimensiones obteniendo los siguientes resultados

Notemos que el tiempo de CPU que se tarda en encontrar un $\sigma_{óptimo}$ que cumpla con el error admisible no aumenta siempre, aunque, se muestra cierta tendencia a aumentar con el número de dimensiones. Esto se puede deber a

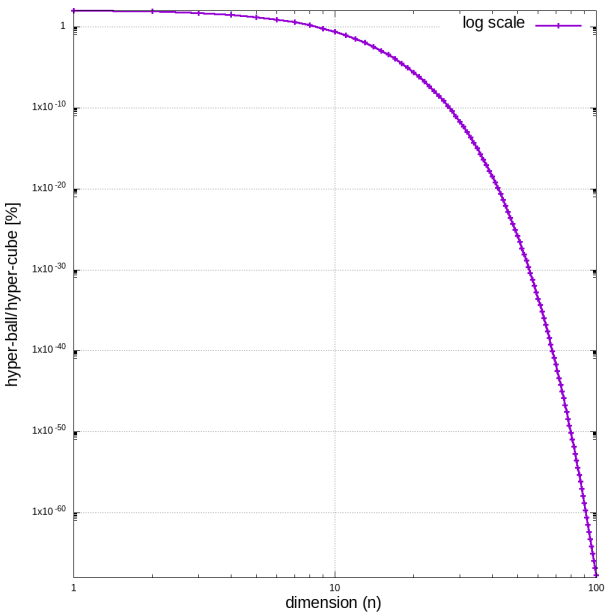

Fig. 4. razon entre volumenes (ball/cube) vs dimensiones

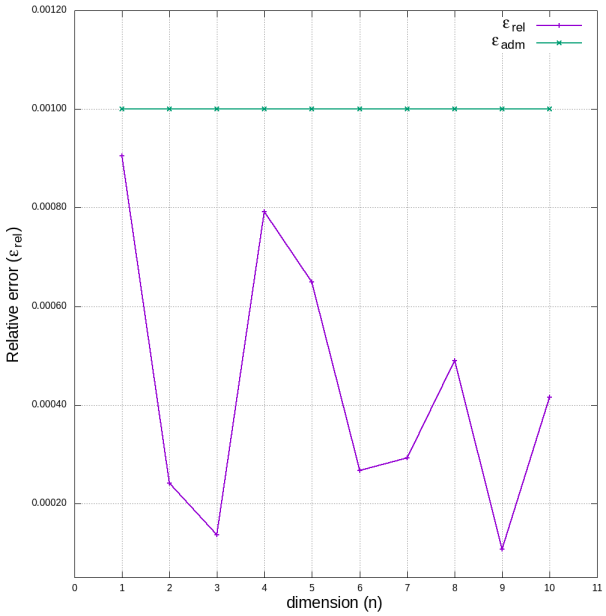

Fig. 6. error relativo vs dimensiones

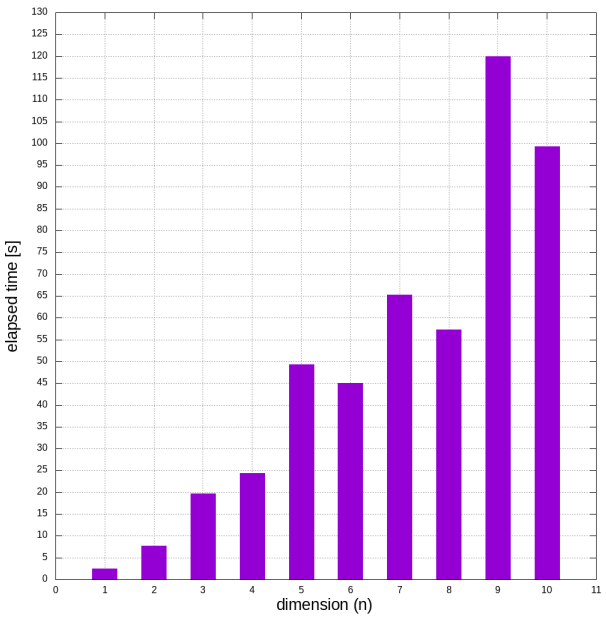

Fig. 5. tiempo de CPU vs dimensiones

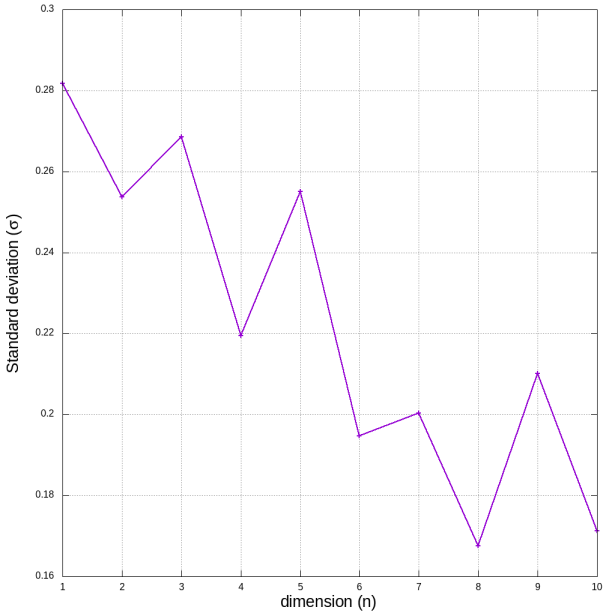

Fig. 7. desviación estandar vs dimensiones

muchos factor, por un lado, notemos que el error relativo es muy variable, si bien, siempre estamos por debajo del error admisible, el $\sigma_{óptimo}$ encontrado no es el mínimo que realmente verificaría, esto es debido a la *ley* de variación del $\Delta\sigma$ con el número de dimensiones que no es exacta. También observando como varía el $\sigma_{óptimo}$ encontrado en función de las dimensiones no vemos una disminución continua como esperaríamos, sino que hay picos y valles, lo cual nos muestra que debemos encontrar una *ley* de variación más precisa para disminuir continuamente el tiempo de CPU, obtener un error relativo más estable y disminuir continuamente el $\sigma_{óptimo}$ al aumentar el número de dimensiones. Cabe aclara que los tiempos de CPU encontrados con la *ley* ad-hoc propuesta se vieron disminuidos notablemente lo que nos pone en evidencia que vamos por buen camino.

Además, para mayor comprensión de lo rápido que disminuye la magnitud del volumen de la hiper-esféra a medida que aumentamos el tamaño de dimensiones se graficó lo siguiente,

donde podemos notar que la función volumen cuenta con un maximal en $n = 5$ y luego disminuye continuamente.

III. Códigos

Repositorio de GitHub

- https://github.com/mendzmartin/fiscomp2022.git

Repositorio GitHub del problema

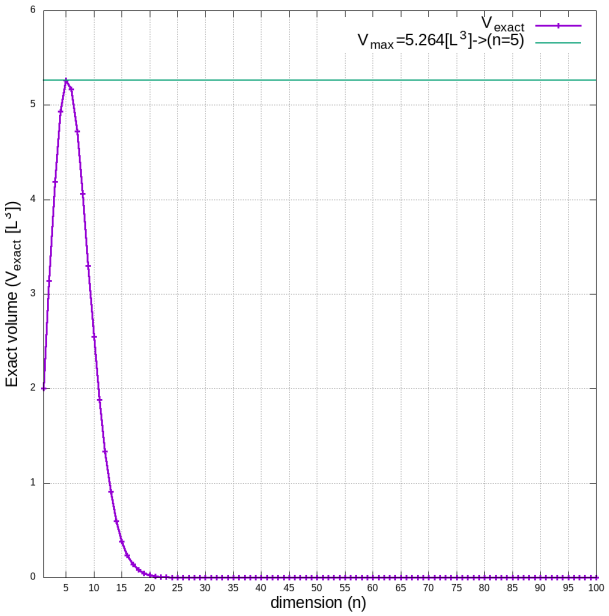

Fig. 8. Volumen exacto vs dimensiones

- https://github.com/mendzmartin/fiscomp2022/tree/main/lab03/prob04

Códigos principales y Makefile

- https://github.com/mendzmartin/fiscomp2022/blob/main/lab03/prob04/code/hyper_sphere.f90
- https://github.com/mendzmartin/fiscomp2022/blob/main/lab03/prob04/code/Makefile