Métodos de Monte Carlo

Entrega 3 - Grupo 01

(Unidad 2, Sesiones 5 y 6) Ejercicios 5.1 y 6.2

Integrantes

Nombre	CI	Correo
Nadia Martínez	5.141045-0	nadia.martinez.petrone@fing.edu.uy
Germán Ouviña	4.823.566-1	german.ouvina@fing.edu.uy

Índice

1. Problema

Entrega 3: Ejercicio 5.1 (GRUPAL):

Se desea estimar el volumen de una región R de $[0,1]^6$ definida por todos los puntos de la hiper-esfera de centro $(0{,}45, 0{,}5, 0{,}6, 0{,}6, 0{,}5, 0{,}45)$ y radio 0,35, que además cumplan las restricciones siguientes: $3x_1 + 7x_4 \leq 5$; $x_3 + x_4 \leq 1$; $x_1 - x_2 - x_5 + x_6 \geq 0$

1. Compartir en el grupo los códigos desarrollados para la parte a, validarlos revisando los códigos, y verificando si las salidas para tamaños de muestra de 10^6 son consistentes. Indicar si se detectaron errores en los mismos, y en ese caso dar los c´odigos corregidos. Elegir uno de los códigos para las partes siguientes, explicar los motivos de la selección.

2. Calcular la cantidad de replicaciones a realizar para garantizar un error menor a $1{,}0 \times 10^{-4}$ con probabilidad 0,95, utilizando el criterio de peor caso de Hoeffding.

3. Utilizando el código elegido en la parte a, y la cantidad de replicaciones definida en el punto anterior, calcular el intervalo de confianza de nivel 0,95 utilizando el criterio de Chebyshev, y el criterio de Agresti-Coull. Comparar el ancho de estos intervalos entre sí y con el criterio de error manejado en el punto previo.

Entrega 3: Ejercicio 6.2 (GRUPAL):

Se desea estimar la integral de la función $x_1 x_2^2 x_3^3 x_4^4 x_5^5$ sobre el hipercubo J^m de dimensión $m = 5$.

1. Revisar los códigos preparados para el ejercicio 6.1, elegir uno de ellos como punto de partida. Sobre esa base, modificarlo para realizar cálculo por Monte Carlo de la integral planteada en el ejercicio 6.2. Realizar 10^6 replicaciones y estimar el valor de ζ. Calcular analíticamente el valor exacto de la integral.

2. En base al valor estimado en la parte a, calcular el número de replicaciones necesario para obtener un error menor a 10^{-4} (con nivel de confianza 0,95).

3. Decimos que un intervalo de confianza cubre el valor exacto cuando este último pertenece al intervalo. Realizar L = 500 experimentos con semillas diferentes, cada uno consistente en estimar por Monte Carlo con el nro. de replicaciones de la parte b el valor de la integral, así como intervalos de confianza de nivel 0,9, 0,95 y 0,99. Para cada nivel de confianza, calcular el nivel de cobertura empírico (en que porcentaje de los 500 experimentos el intervalo de confianza calculado cubrió el valor exacto). Discutir los resultados, comparando la cobertura empírica con la especificada.

2. Solución

2.1. Descripción

2.1.1. Ejercicio 5

Para este ejercicio, se compararon tanto el código escrito como los resultados obtenidos.

Respecto al código, se realizaron las siguientes observaciones:

- El código de Nadia tiene un enfoque vectorial, generando los n puntos desde un comienzo y realizando operaciones vectoriales para comprobar si se encuentran en la región y acumular los valores correspondientes.
- El código de Nadia es un poco más escueto, aprovechando mejor las ventajas de la vectorización de la biblioteca *NumPy* y la manipulación natural de tuplas dentro de *Python*.
- El código de Germán tiene un enfoque iterativo, iterando n veces pero generando un punto por iteración, siguiendo la idea del pseudocódigo presentado en las sesiones de teórico.
- El código de Germán es un poco más extenso, trabajando coordenada a coordenada en lugar de agrupar en 6-uplas para cada punto. Por otra parte, agrega funcionalidades como parametrización al ejecutar (la posibilidad de proveer distintos valores por parámetro además de n, como la opción de ejecutar el método con o sin las restricciones extra de la región, y la opción de realizar una única ejecución del método o múltiples que generen tablas y gráficas comparativas).

Respecto a los resultados, se realizaron las siguientes observaciones:

- El código de Nadia llega al resultado para $n = 10^6$ en un tiempo aproximado a 4 segundos, siendo adaptable a mayores tamaños de n.
- El código de Nadia ocupa un máximo de memoria para la estructura de puntos aleatorios de $size(float) * 6 * n$, volviéndose subóptimo para mayores tamaños de n debido al crecimiento exponencial del espacio en memoria utilizado.
- El código de Germán llega al resultado para $n = 10^6$ en un tiempo aproximado de 11 segundos, volviéndose subóptimo para mayores tamaños de n debido al crecimiento exponencial del tiempo de ejecución requerido.
- El código de Germán ocupa un máximo de memoria para la estructura de puntos aleatorios de $size(float) * 6$, reciclando ese espacio por iteración y siendo adaptable para mayores tamaños de n.
- Ambos códigos devuelven el exacto mismo resultado si se emplea la misma semilla en ambos.

Teniendo en cuenta que ambos códigos llegan al mismo resultado exactamente y ambos cuentan con pros y contras, en primera instancia se decidió elegir el código de Nadia, ya que ejecuta más rápido. Si bien el código iterativo es más intuitivo, el código vectorizado es mucho más rápido y para mayores tamaños de n eso se vuelve clave.

No obstante, al intentar ejecutarlo para valores muy grandes de n hubo problemas por memoria. La solución a la que se llegó fue emplear lo mejor de ambos mundos: se mezclaron ambos códigos en uno, utilizando el enfoque vectorizado pero en *batches*, ejecutando $\frac{n}{batch}$ iteraciones donde se sortean $batch$ puntos.

En la sección 2.2.1 se entra en mayor detalle sobre las características del código. En la sección 3.2.1 se detallan los resultados obtenidos para las partes b y c pedidas en la letra.

2.1.2. Ejercicio 6

Para este ejercicio, se comparo el código escrito por ambos para el ejercicio individual y, de manera similar al ejercicio 5, se llegó a una mezcla entre los mejores aspectos de ambos. En esta instancia, ambos algoritmos utilizan enfoques iterativos, llegan a los mismos resultados y en general siguen los mismos lineamientos.

Teniendo en cuenta las similitudes con ejercicios anteriores, se utilizó de esquema base el programa utilizado en el ejercicio 5, con la intención de aprovechar funcionalidades para generar tablas y gráficas, así como para calcular intervalos de confianza y cantidad de réplicas óptima para ciertos valores de error y confianza.

Cabe destacar que, a diferencia del ejercicio 5, el método para calcular intervalos de confianza empleado fue el de *aproximación normal*, en lugar de los métodos de *Chebyshev* y *Agresti-Coull*. Para el cálculo de cantidad óptima de réplicas, también se utilizó el método de *aproximación normal* en lugar del criterio de *Hoeffding*. Ambos métodos integran el uso del estimador de varianza puntual de la función a calcular a diferencia de los métodos anteriores que son agnósticos a los resultados obtenidos durante el experimento.

En la sección 2.2.2 se entra en mayor detalle sobre las características del código. En la sección 3.2.2 se detallan los resultados obtenidos para las partes a, b y c.

2.2. Pseudocódigo

2.2.1. Ejercicio 5

Procedimiento EstimacionMonteCarloVolumen (int n, int b, region R, real δ)
Parámetros de entrada:

- n - tamaño de la muestra
- b - tamaño de batch
- R - región a calcular el volumen
- δ - nivel de confianza

Parámetros de salida:

- $\hat{X}$ - estimador del volumen
- $\hat{V}$ - estimador de la desviación estándar
- ω_1, ω_2 - intervalo de confianza para $(1 - \delta)$

Pseudocódigo:

1. $X, \hat{X}, \hat{V} = 0$
2. For $i = 0, ..., \frac{n}{b} + 1$ do
   a) Sortear coordenadas de b puntos $X_i = X_{bi}, X_{bi+1}, ..., X_{bi+b-1}$ siguiendo la distribución $U(0, 1)$ por coordenada
   b) Si $X_j \in R \rightarrow X = X + 1$ para cada X_j en la matriz X_i
3. $\hat{X} = X/n$
4. $\hat{V} = \hat{X}(1 - \hat{X})/(n - 1)$
5. $\omega_1, \omega_2 = I_1(X, n, \delta), I_2(X, n, \delta)$

El anterior pseudocódigo bosqueja un método de monte carlo para el cálculo de volumen de una región, siguiendo las pautas del pseudocódigo presentado en clase. Extendiendo el código de la entrega anterior, se agrega la noción de la ejecución por batches, definiendo un tamaño de batch b y operando con una matriz de b puntos de manera vectorial, en n/b iteraciones. Esto permite agilizar el proceso sin comprometer el uso de memoria. Por otra parte, se agrega el cálculo del intervalo de confianza según el método indicado y el uso de X como acumulador separado al estimador $\hat{X} = X/n$.

2.2.2. Ejercicio 6

Procedimiento EstimacionMonteCarloIntegral (int n, funcion f, real δ, real ϵ)
Parámetros de entrada:

- n - tamaño de la muestra
- f - función a calcular su integral
- δ - nivel de confianza
- ϵ - error máximo

Parámetros de salida:

- $\hat{X}$ - estimador de la integral
- $\hat{V_f}$ - estimador de la varianza de la función f
- $\hat{V_X}$ - estimador de la varianza del estimador puntual $\hat{X}$
- ω_1, ω_2 - intervalo de confianza para $(1 - \delta)$
- n_N - cantidad óptima de replicaciones que garantiza un error máximo de ϵ con un nivel de confianza de $1 - \delta$

Pseudocódigo:

1. $X, \hat{X}, \hat{V_f}, \hat{V_X} = 0$
2. For $i = 1, ..., n + 1$ do
   a) Sortear coordenadas de un punto p siguiendo la distribución $U(0, 1)$ por coordenada
   b) Evaluar $f(p)$
   c) Acumular $X = X + f(p)$
   d) Si $i > 1 \rightarrow \hat{V_f} = \hat{V_f} + (1 - \frac{1}{i})(f(p) - \frac{X}{i-1})^2$
3. $\hat{X} = X/n$
4. $\hat{V_f} = \hat{V_f}/(n - 1)$
5. $\sqrt{\hat{V_X}} = \sqrt{\hat{V_f}/n}$
6. $\omega_1, \omega_2 = I_1(\hat{X}, \sqrt{\hat{V_X}}, \delta), I_2(\hat{X}, \hat{V_X}, \delta)$
7. $n_N = n_N(\epsilon, \delta, \sqrt{\hat{V_f}})$

El anterior pseudocódigo bosqueja un método de monte carlo para el cálculo de la integral de una función, siguiendo las pautas del pseudocódigo presentado en clase. Como cambio básico, se evalúan los puntos sorteados en la función objetivo f y se acumula dicho valor en lugar de acumular 1 por cada punto sorteado que pertenece a la región (como se hacía al calcular un volúmen). A su vez, se modifica el cálculo de la varianza y se distingue entre dos valores: $\hat{V_f}$ para la varianza de la función y $\hat{V_X}$ para la varianza del estimador en sí. Esto se debe a que ambos valores son útiles para distintos datos de interés que se calculan al final, concretamente el intervalo de confianza (ω_1, ω_2) y la cantidad óptima de replicaciones n_N, ambos métodos siguiendo la aproximación normal que emplean las varianzas $\hat{V_X}$ y $\hat{V_f}$ respectivamente. Cabe destacar que el cálculo del intervalo de confianza en esta instancia si utiliza el valor del estimador $\hat{X}$ en lugar del acumulador X.

2.3. Código

El código para solucionar el problema se encuentra adjunto en un archivo *.zip* junto al informe, y se encuentra disponible en el repositorio https://github.com/gouvina-fing/fing-mmc2025.

El mismo fue desarrollado en *Python*, cuenta con instrucciones de como instalarlo y correrlo, y ofrece la posibilidad de generar tablas y gráficas comparativas siguiendo la consigna de ambos ejercicios, automáticamente ejecutando el método para los distintos valores de n mencionados.

También existe un programa aparte para el cálculo de los tamaños de muestra pedidos en ambos ejercicios, el cual permite obtener el tamaño de muestra n que garantiza un error ϵ con nivel de confianza $(1 - \delta)$.

Los programas mencionados son los siguientes, y se ejecutan con el siguiente conjunto de parámetros en el órden correspondiente:

- python lab3_ej5_calculo_muestra.py con parámetros:
  - Tipo de ejecución - Por defecto *simple*. Puede ser también *búsqueda*
  - Error ϵ - Por defecto 0,01
  - Nivel de confianza δ - Por defecto 0,05
  - Algoritmo - Por defecto n_C (Chebyshev). Puede ser también nN (TLC) o nH (Hoeffding)
- python lab3_ej5_calculo_volumen.py con parámetros:
  - Tamaño de muestra n - Obligatorio
  - Tamaño de batch b - Por defecto 1000
  - Nivel de confianza δ - Por defecto 0,05
  - Intervalo - Por defecto *chebyshev*. Puede ser también *agresti-coull*
  - Tipo de ejecución - Por defecto *simple*. Puede ser también *búsqueda*
- python lab3_ej5_calculo_integral.py con parámetros:
  - Tamaño de muestra n - Obligatorio
  - Error ϵ - Por defecto 0,01
  - Nivel de confianza δ - Por defecto 0,05
  - Tipo de ejecución - Por defecto *simple*. Puede ser también *cobertura* o *búsqueda*

3. Experimentación

3.1. Especificaciones

A continuación, se desglosan las especificaciones técnicas utilizadas durante la ejecución:

- **Procesador:** 11th Gen Intel(R) Core(TM) i5-11400F @ 2.60GHz
- **Memoria RAM:** 16.0 GB
- **Sistema Operativo:** Windows 10
- **Semilla:** 42
- **Tamaños de muestra:** 10^4, 36413, 10^5, 10^6, 10^7, 10^8, 10^9, 184443973

3.2. Resultados

3.2.1. Ejercicio 5

La consigna de este ejercicio cuenta con dos partes en lo que refiere a la experimentación: en primera instancia, obtener una cantidad de replicaciones n para cierto margen de error con cierto nivel de confianza. En segunda instancia, ejecutar el algoritmo para ese tamaño de muestra encontrado y calcular el intervalo de confianza según el método de Chebyshev y el de Agresti-Coull.

Para la primera parte, se utilizó el criterio de peor caso de Hoeffding y valores de error y nivel de confianza de $\epsilon = 1{,}0 \times 10^{-4} = 0{,}0001$ y $1 - \delta = 0{,}95$ respectivamente. La cantidad de replicaciones obtenida fue de $n = 184443973$, un valor en el orden de 10^8, bastante mayor a los empelados en anteriores experimentos. Esta fue la principal razón detrás de la idea de fusionar ambos códigos para tener un enfoque tanto iterativo como vectorial, procesando el espacio muestral en batches.

Para la segunda parte, además de ejecutar el algoritmo para el valor de n obtenido en la primera parte, se generaron valores para $n = 10^4, 10^5, 10^6, 10^7, 10^8$, con el objetivo de comparar los comportamientos tanto del estimador y de la desviación estándar, como de los intervalos de confianza generados por ambos métodos. Se comenzó con $n = 10^4$ debido a que en el anterior experimento, valores de n menores no retornaron resultados relevantes.

A continuación, se adjunta la tabla comparativa marcando en rojo los valores de interés según la consigna:

N° de iteraciones	Estimador ($\hat{X}$)	Desviación ($\hat{V}$)	IdeC (Chebyshev)	IdeC (Agresti-Coull)	Tiempo (s)
10^4	0.0004000000	0.0001999700	0.0000583618 0.0027360494	0.0001152606 0.0010684306	0.16
10^5	0.0002600000	0.0000509838	0.0001110070 0.0006088491	0.0001756504 0.0003827428	1.62
10^6	0.0002790000	0.0000167010	0.0002136403 0.0003643482	0.0002480740 0.0003137653	14.06
10^7	0.0002757000	0.0000052500	0.0002531995 0.0003001994	0.0002655986 0.0002861853	159.15
10^8	0.0002818400	0.0000016786	0.0002744325 0.0002894474	0.0002785691 0.0002851493	1568.21
184443973	0.0002819610	0.0000012362	0,0002764863 0,0002875440	0,0002795483 0,0002843944	2941.14

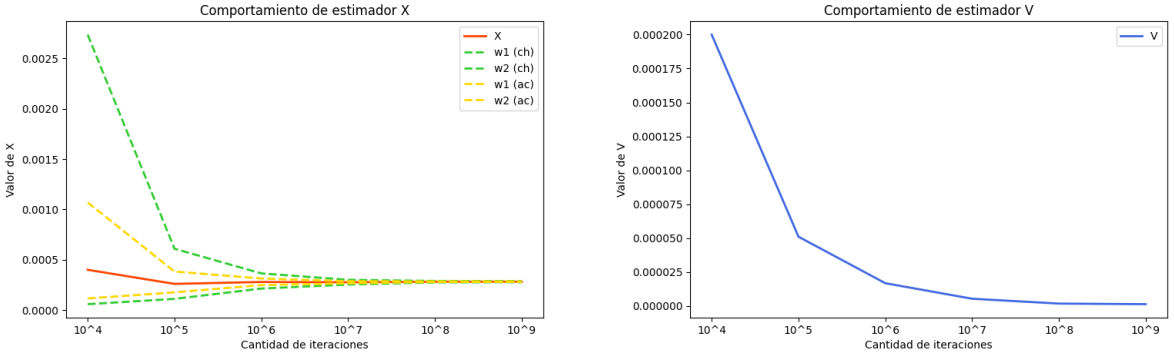

Figura 3.1: Comportamiento de estimadores en cálculo de volumen

Tanto de la tabla como de las gráficas se puede observar el comportamiento esperado: los estimadores de volumen y desviación tienen comportamiento asintótico como de costumbre y los intervalos de confianza acotan el valor del estimador de volumen, acercandose cada vez más para mayor cantidad de iteraciones. El intervalo de Agresti-Coull converge mejor y más rápido que el de Chebyshev como se esperaba.

3.2.2. Ejercicio 6

La consigna de este ejercicio cuenta con tres partes en lo que refiere a la experimentación: en primera instancia, ejecutar el algoritmo para $n = 10^6$ y calcular un intervalo de confianza utilizando la aproximación normal. En segunda instancia, obtener una cantidad de replicaciones n para cierto margen de error con cierto nivel de confianza, y ejecutar el algoritmo para ese tamaño de muestra. En tercera instancia, realizar 500 experimentos con distintas semillas y distintos niveles de confianza para calcular el nivel de cobertura empírico en cada caso.

Para las primeras dos partes, se ejecutó el algoritmo con $n = 10^6$ y se utilizó el criterio de aproximación normal para calcular n_N, con valores de error y nivel de confianza de $\epsilon = 1{,}0 \times 10^{-4} = 0{,}0001$ y $1 - \delta = 0{,}95$ respectivamente. La cantidad de replicaciones obtenida fue de $n = 36413$, un valor en el orden de 10^4, bastante pequeño en comparación al obtenido en el ejercicio anterior. Además de ejecutar el algoritmo para el valor de n_N obtenido se generaron valores para $n = 10^4, 10^5, 10^6$, con el objetivo de comparar los comportamientos tanto del estimador y de las varianzas, como de los intervalos de confianza generados. Se comenzó con $n = 10^4$ debido a que en el anterior experimento, valores de n menores no retornaron resultados relevantes.

A continuación, se adjunta la tabla comparativa marcando en rojo los valores de interés según la consigna:

N° de iteraciones	Estimador ($\hat{X}$)	Varianza ($\hat{V}_f$)	Desviación ($\sqrt{\hat{V}_X}$)	IdeC (Normal)	Tiempo (s)
10^4	0.0013806770	0.0000842387	0.0000917816	0.0012007883 0.0015605657	0.10
36413	0.0013705249	0.0001005418	0.0000525467	0.0012675354 0.0014735145	0.36
10^5	0.0014015111	0.0001005035	0.0000317023	0.0013393758 0.0014636464	1.02
10^6	0.0013932313	0.0000936053	0.0000096750	0.0013742687 0.0014121939	10.51

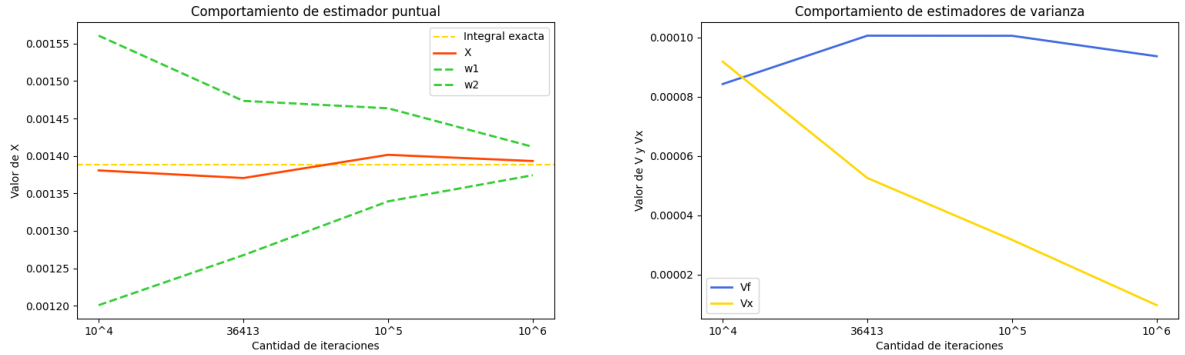

Figura 3.2: Comportamiento de estimadores en cálculo de integral

Tanto de la tabla como de las gráficas se pueden observar diversos fenómenos. Como suele suceder, el valor del estimador puntual se encuentra siempre acotado por el intervalo de confianza, el cual parece converger hacia el valor en cuestión al aumentar la cantidad de iteraciones. Resulta coherente asumir eso, ya que en este caso en particular, el método empleado para calcular el intervalo de confianza parte exactamente del valor del estimador. Una observación no menor es que, en comparación a otros ejercicios, el valor del estimador oscila en lugar de tener una tendencia estrictamente creciente o decreciente. Sin embargo, al tratarse de un método estocástico, resulta aceptable observar tal comportamiento. Particularmente, para el valor de $n = 36413$, el valor del estimador obtenido fue de $\hat{X} = 0{,}0013705249$, el cual se encuentra dentro del rango de error $\epsilon = 0{,}0001$.

Para la útlima parte, se ejecutó el algoritmo con $n = 36413$, para $L = 500$ experimentos, utilizando semillas dentro del rango $[1, 500] \cap \mathbf{Z}$. Para cada experimento realizado, se comprobó si el valor exacto de la integral (calculado analíticamente como $1/720 = 0{,}00139$ aproximadamente) se encontraba en el rango del intervalo de confianza calculado. A través de ese índice se obtuvo la cobertura empírica.

A continuación, se adjunta la tabla comparativa para cada valor de $1 - d$ empleado y la cobertura empírica obtenida:

Nivel de confianza $(1 - d, \%)$	**Cobertura empírica (%)**
90 %	91,6 %
95 %	96,8 %
99 %	99 %

En las siguientes páginas, se adjuntan dos grupos de gráficas para cada valor de δ en el rango $[0{,}90, 0{,}95, 0{,}99]$.

En el primer trío de gráficas, se observa el valor obtenido del estimador para cada uno de los 500 experimentos realizados, así como las curvas que representan las cotas del intervalo de confianza. Es difícil apreciar en las 3 imagenes separadas, pero a medida que δ disminuye, el comportamiento del intervalo de confianza y de los valores estimados parece ser menos errático, manteniendo la línea punteada horizontal (valor exacto de la integral) dentro del rango con cada vez mayor precisión.

En el segundo trío de gráficas, se observan todos los valores del estimador para cada experimento con su valor fijo alrededor del valor exacto de la integral, excepto para aquellos cuyo intervalo de confianza no logró acotar correctamente dicho valor exacto (para esos experimentos, el valor del estimador se graficó en 0). A medida que δ disminuye, se observa como cada vez menos valores del estimador quedan fuera, contando unos pocos puntos por debajo del cúmulo (para el último caso, siguiendo los cálculos de la cobertura empírica, solo un 1 % queda por debajo, es decir, 5 valores).

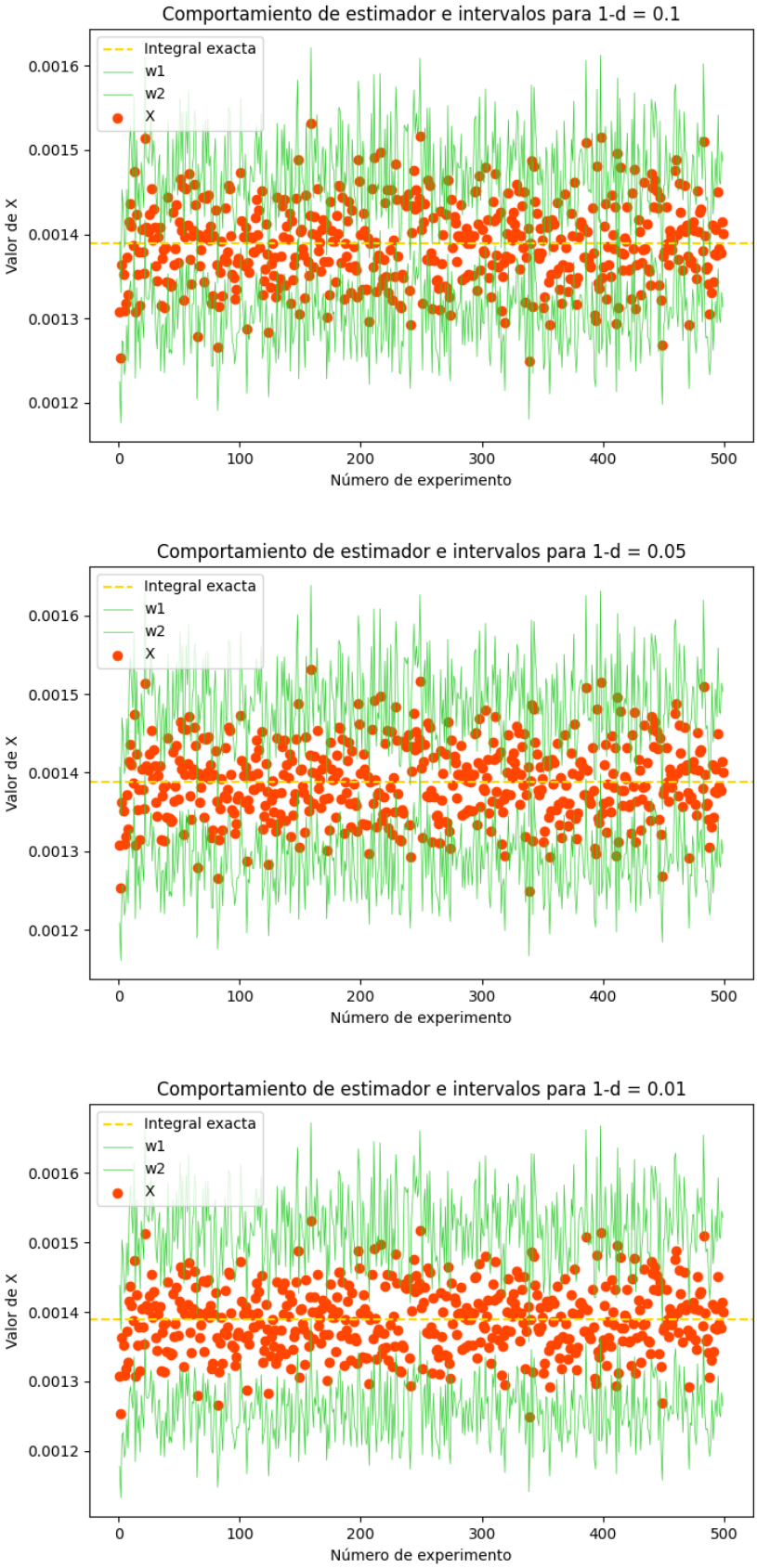

Figura 3.3: Comportamiento de estimadores e intervalos de confianza en 500 experimentos

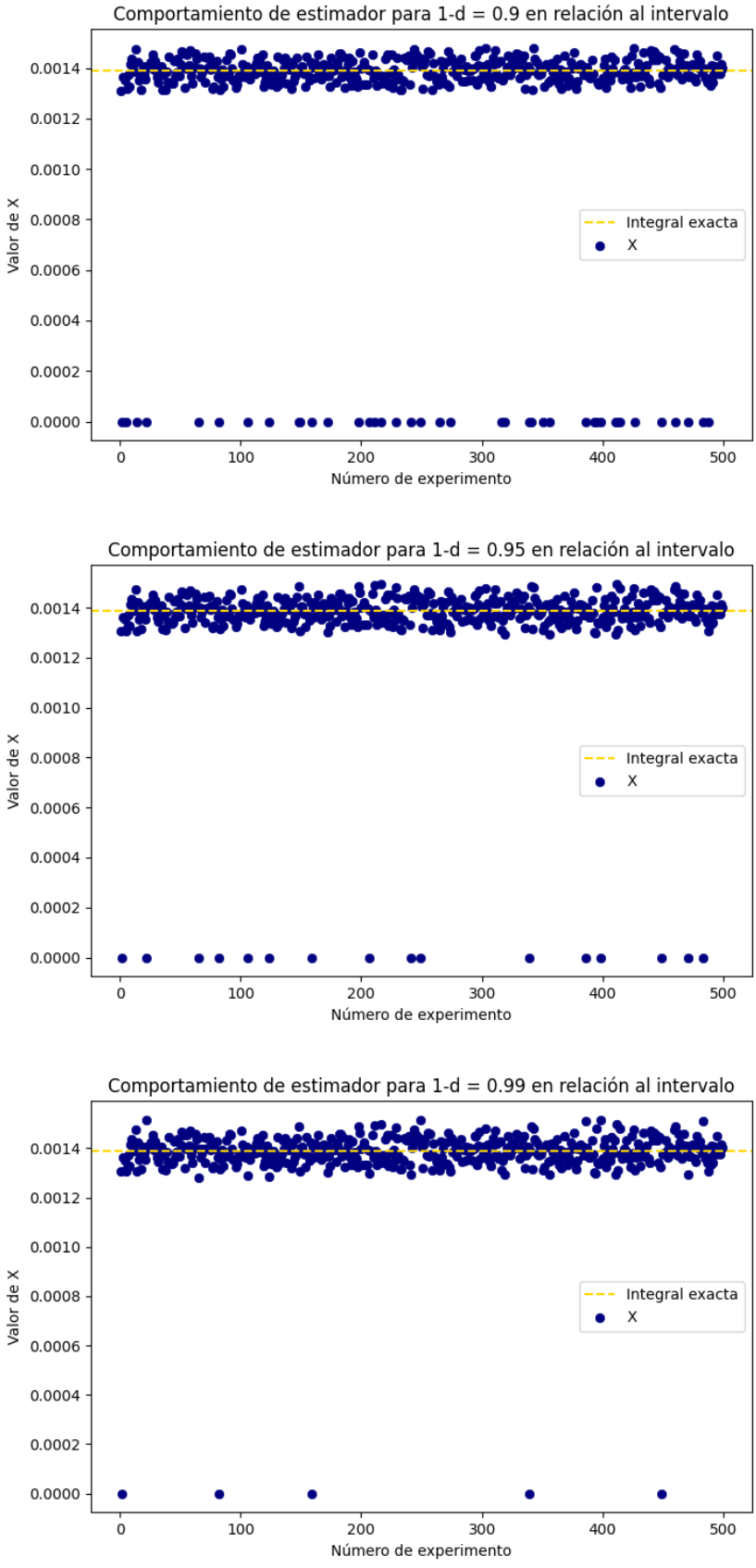

Figura 3.4: Valores del estimador que son cubiertos empíricamente por los intervalos en 500 experimentos