Método de Monte Carlo para integrales en múltiples dimensiones
Física Computacional 1

Roberto Vázquez - A01196927, Eugenio Butler - A00819945,
Carla López - A00822301, and Rubén Casso - A01196975

ABSTRACT: Se utilizó el método de integración de Monte Carlo para distintos ejemplos, con el fin de aprender a implementarlo en problemas de diferentes contextos. Se abordaron integrales sencillas (función Gamma) y multidimensionales (volúmenes de hiperesferas). Se interrpeta el alcance y utilidad del método, a través de un análisis de errores relativos.

I. INTRODUCCIÓN

A lo largo del curso de *Física Computacional I*, se discutió en diversas ocasiones la importancia de los métodos numéricos para la aproximación y manipulación de funciones, en particular aquellas que son difíciles de resolver de manera analítica. Una gran parte de los métodos vistos en clase eran de integración, con el objetivo de encontrar una solución numérica a integrales y, de ser posible, compararlo con su valor real. En su mayoría, consistían en seccionar la función dada de distintos modos y conseguir una mejor aproximación a la respuesta a medida que los intervalos alcanzan dimensiones de menor tamaño o se aumente el número de iteraciones del método.

Un conjunto de ejemplos de estos algoritmos es la familia de funciones Newton-Cotes, a la cual pertenecen los tipos de integración interpolatoria como la trapezoidal y las reglas de Simpson. Se presentaron tambien los métodos de Romberg para, con recursividad, conseguir mejores resultados; y algunos más complejos como la cuadratura Gaussiana, donde ya están definidos los pesos de las funciones, según el número de puntos utilizados. Hasta ese entonces, los métodos eran, en gran parte, sencillos de programar, y se tenía la oportunidad de escoger aquel que más se adaptara a la situación que se intentaba resolver. Sin embargo, al tratar de escalar las técnicas aprendidas a dimensiones mayores, surge un problema de gran escala: manipular matrices de n dimensiones resultó ser muy pesado computacionalmente y no tan accesible como fue hacerlo para integrales en dos dimensiones. Con un poco de curiosidad (herramienta indispensable en la física) nace la pregunta: ¿existirá alguna otra manera de aproximar integrales distinta a las vistas en clase?

La respuesta es muy sencilla: sí existen, y una de ellas es un método para evaluar integrales basado en probabilidad y estadística llamado Monte Carlo, nombrado así por los famosos casinos de la ciudad homónima. Esta técnica se llegó a mencionar brevemente durante una de las sesiones, por lo que la motivación de utilizar Monte Carlo en el proyecto reside precisamente en aprender un nuevo método por nuestra cuenta y aplicar esta herramienta en ejemplos sencillos para comparar su efectividad.

A continuación, se presentará la implementación y los resultados de este método, utilizado para resolver integrales en dos y más dimensiones. Igualmente, se redactarán las limitaciones encontradas al tratar de optimizar los resultados y, finalmente, la valoración final del método por parte del equipo.

II. MÉTODO DE MONTE CARLO

Nacido del deseo de obtener probabilidades en juegos de azar, se conoce como método de Monte Carlo al conjunto de algoritmos que se enfocan en sacar muestras aleatorias, con el fin de obtener resultados númericos. Así, la belleza de la técnica recae en usar la aleatoreidad para solucionar no solo problemas estocásticos (como las muestras tomadas), sino también aquellos que son determinísticos en principio.[1]

Dada su flexibilidad, este método se utiliza en distintas áreas de las ciencias y con variados fines. Además de funcionar para integración, es posible adaptarlo a problemas de optimización y de probabilidad, y se vuelve también relevante en la simulación de fenómenos con un alto grado de incertidumbre. En física, específicamente, funciona para modelar dinámicas moleculares, diseñar detectores, estudiar la evolución de las galaxias, y forma la base para hacer pronósticos meteorológicos, por mencionar algunos ejemplos. La razón se debe a que estos modelos suelen utilizar funciones multivariables; y como ya se ha mencionado, Monte Carlo es particularmente útil con el trabajo de mayores dimensiones, lo cual llega a ser una limitante en otros métodos numéricos.

A. Descripción de la técnica

A grandes rasgos, el método Monte Carlo trabaja con muestras aleatorias de números y las "dispara" en un espacio dado sobre el cual está inscrita la función que se busca integrar. Visualmente, la aplicación del método se ve algo como la gráfica siguiente:

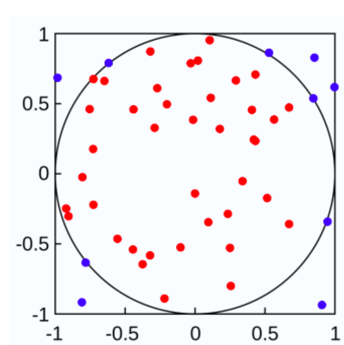

Figura 1. Representación gráfica del método Monte Carlo para calcular área de un círculo

Una analogía sencilla que ayuda a comprender cómo funciona el método es esta: Se tiene un tablero sobre el cual se encuentra una diana (o blanco de tiro) circular. Si se empiezan a tirar dardos para intentar dar en la diana, algunos caerán dentro y otros caerán sobre el tablero, fuera del blanco. Ahora, se repiten los disparos miles, o incluso millones de veces, hasta que la diana se llena de dardos. Entonces, si se conoce el área del tablero grande, así como la cantidad de dardos que cayeron dentro y la cantidad que cayeron fuera del blanco, es posible conocer, mediante simples proporciones, el área de la diana.

En el caso de la Figura 1, hay un círculo de radio unitario, dentro de un cuadrado de 2 unidades de largo, de tal forma que el círculo queda completamente inscrito en él. Los puntos rojos son los experimentos (dardos) que cayeron dentro del círculo (diana), mientras que los puntos azules son aquellos que cayeron fuera del círculo (en el tablero). Es fácil notar que, si se obtuvieran las proporciones para conocer el área del círculo, la aproximación sería muy pobre y se alejaría del valor real. Así, se concluye que la validez del método Monte Carlo incide en la ley de los grandes números, la cual dicta que al repetir un experimento múltiples veces, el promedio de los resultados irá convergiendo al valor esperado.[2] Como se suponía, el número de iteraciones se vuelve una variable relevante a considerar para el trabajo, y su uso debido marca la diferencia entre si el método funciona o no y a qué grado.

B. Algoritmo

Una vez comprendida la técnica y con todo lo aprendido hasta ahora, se puede proseguir con la construcción del algoritmo que se utilizará para todo el desarrollo del proyecto. A continuación se describe, paso a paso, dicho procedimiento:

1. Primeramente, se define un espacio conocido en donde existe la función y se calcula su área, volumen, o análogo. Por ejemplo, para el caso bidimensional, se define el cuadrado donde vive la función; para el tridimensional, se define el cubo donde vive la función, y así subsecuentemente.
2. Se generan puntos aleatorios (experimentos) en dicho espacio definido.
3. Se cuentan los puntos que caen dentro de la función, denominados *INRANGE* en este trabajo.
4. Se calcula la rázon entre todos los puntos *INRANGE* y los puntos totales generados.
5. Y por último, se multiplica la rázon obtenida por el área/volumen del espacio total debido a la relación:

$$\frac{Puntos\ INRANGE}{Puntos\ TOTALES} = \frac{Integral}{Volumen}$$

Para probar el procedimiento anterior, se decidió comenzar con un caso bidimensional sencillo (además del cálculo del área del círculo ya mencionado), antes de escalar a mayores dimensiones. Por lo tanto, se buscó utilizar alguna integral manejable y bien comportada. De entre todas las opciones disponibles, se escogió la función gamma, cuyo valor exacto se obtiene fácil, pues ya está programada en *Matlab*. Así, la gamma posteriormente podría ser utilizada dentro de otras funciones conocidas para analizar la propagación del error numérico en su cálculo con el método de Monte Carlo.

III. DOS DIMENSIONES: FUNCIÓN GAMMA

La función gamma es una herramienta matemática que generaliza la operación de factorial para números complejos, racionales y negativos.[3] Por definición, se calcula mediante la integral siguiente:

$$\Gamma(z) = \int_0^\infty t^{z-1} e^t dt$$

Y para $n \in \mathbb{Z}^+$ se cumple la siguiente relación:

$$\Gamma(n) = (n-1)!$$

A partir de la definición integral de la gamma, se programó en *Matlab* la función, y los parámetros de la misma utilizando la aleatoreidad que caracteriza al método Monte Carlo, como se muestra a continuación:

```
xp = rand(1,n);
yp = rand(1,n)*max(G);
ygamma = ((1-xp)./xp).^(z-1).*
    exp(-((1-xp)./xp))./xp.^2;

INRANGE=yp<=ygamma;

Ratio=sum(INRANGE)/n;
AreaSquare=x(end)*max(G);
AreaGamma=AreaSquare*Ratio;
```

En el extracto de código anterior, primeramente se crean dos vectores, *xp* y *yp* que contienen valores aleatorios, para que de esta forma se generen los puntos en el espacio. El vector *yp* se genera de la misma manera que *xp*, pero se multiplica por $max(G)$ para que se distribuyan los puntos en todo el espacio donde vive la función. Después, se calcula propiamente la función gamma con dichos vectores, y se cuentan en la variable booleana *INRANGE* todos aquellos puntos en *yp* que se encuentren dentro de la función.

Ya teniendo esta cifra, tal y como se explicó en la sección anterior, el método de Monte Carlo consiste únicamente en sacar la razón entre los aciertos y el número de puntos totales, y multiplicar por el volumen (o en este caso, área) del espacio ya definido. En la última línea del código, se almacena en la variable *AreaGamma* el resultado final de la integral.

A. Mejora de resultados: Histograma

Mientras se revisaban los resultados de la función gamma programada anteriormente, nos dimos cuenta de algo importante: Dado que el método de Monte Carlo emplea número aleatorios, cada vez que se corre el programa arroja un valor distinto para el cálculo que está realizando. Después de revisar la literatura,[4] se decidió correr el algoritmo múltiples veces como medida para incrementar la exactitud en las soluciones.

Para obtener una representación visual de la variación de los resultados, se realizó un histograma, donde se graficaban cada una de las evaluaciones de la función gamma. En la siguiente imagen, se encuentran los resultados de 1000 iteraciones de la gamma para $z = 4$, que, como sabemos según la definición, es igual a 3!=6.

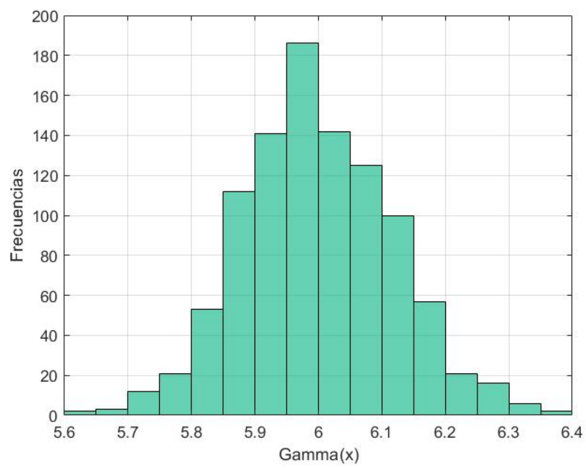

Figura 2. Histograma de valores de la función Gamma evaluada en $z = 4$

Algo que resultó evidente y muy interesante al graficar en repetidas ocasiones los valores contra su recurrencia en un histograma, es que los valores se asemejan a una distribución normal. Y entre más experimentos se realicen con Monte Carlo, más acertados son los resultados, haciendo que el área se aproxime a una delta de Dirac, encendiéndose solo en 6, que es el valor analítico de la función gamma para $z = 4$. Y entre más repeticiones de Monte Carlo se realicen, más confiable es quedarnos con la media del histograma.

Tomando todo esto en cuenta, es posible utilizar como valor representativo de los datos su promedio a través de la función *mean()*. Otra manera de interpretar los datos sería calcular su distribución estándar y formar un intervalo de confianza. Para los datos mostrados en la figura 2, la desviación estándar es de 0,1262 y el intervalo de confianza, con un 90 % de seguridad, es [5.7977,6.2117].

Revisando literatura, se encontró que existen muchas metodologías diferentes que adecúan distribuciones de probabilidad específicas a los espacios aleatorios generados. Esto serviría para ciertos problemas con contextos conocidos, pero para fines del proyecto presentado, se decidió trabajar con distribuciones uniformes.

B. Función Bessel

Esta función, además de servir como pretexto para calcular una integral mediante el método de Monte Carlo, también aparece en otro tipo de funciones, por ejemplo en la solución a la ecuación de Laplace en coordenadas cilindrícas, por lo que reciben el nombre de harmónicos cilíndricos. Las funciones de Bessel son el conjunto de soluciones de la ecuación diferencial de Bessel.

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - \alpha^2)y = 0$$

Las soluciones más importantes son cuando α es un entero o medio entero.

Las funciones Bessel de primer tipo son definidas como una serie alrededor de $x = 0$, obtenida despúes de aplicar el método de Frobenius a la ecuación de Bessel:

$$J_\alpha(x) = \sum_{m=0}^{\infty} \frac{(-1)^m}{m!\Gamma(m+\alpha+1)} \left(\frac{x}{2}\right)^{2m+\alpha} \quad (1)$$

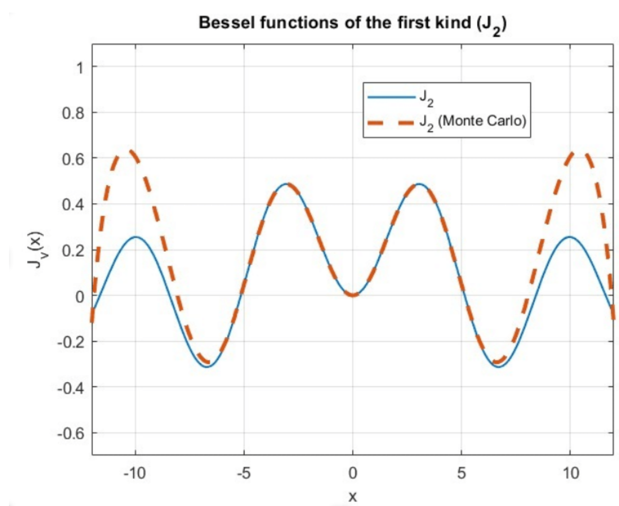

Figura 3. Comaparación gráfica de la Bessel calcualda vs la función de Matlab

En la Figura 3 se observa la función Bessel del paquete de Matlab comparada con la función Bessel en la que la función Gamma se calcula usando el método de Montecarlo. Se observa una región de convergencia, aproximadamente, donde $|x| \leq 6$. La diferencia que se genera en puntos fuera de este dominio se debe a la variabilidad de los cálculos del valor de la Gamma, que aumenta conforme se incrementa el argumento de la función. Entonces para potencias grandes de x (en la función Bessel), el coeficiente no es exacto y el error se aprecia más cuando nos alejamos del cero.

Es sencillo notar que la lógica del procedimiento no es muy compleja, por lo que se investigo el uso de Monte Carlo para problemas en mayores dimensiones.

IV. N-DIMENSIONES: HIPERESFERAS

Con el propósito de escalar el método a integrales de mayores dimensiones, se calcularon los volúmenes de esferas en n dimensiones, debido a que existen en la literatura resultados analíticos con los cuales comprobar nuestro algoritmo. Una hiperesfera es el anólogo de una esfera a diferentes dimensiones.

- Una esfera-0 es un par de puntos {c-r, c+r}
- Una esfera-1 es un circulo (dos dimensiones)
- Una esfera-2 es una esfera en tres dimensiones
- Una esfera-3 es una esfera en cuatro dimensiones
- Una n-esfera se denomina "glone"

En general, se definen de la siguiente manera:

$$S^n = \{x \, \epsilon \, \mathbb{R}^{n+1} \; : \; ||x|| \; = \; r\}$$

Con el fin de verificar el correcto funcionamiento del código, se programó primeramente el método para una esfera-2, ya que esta es la dimensión más alta en la que se puede obtener una representación gráfica. Recordando que, para propósitos de este trabajo, el radio es unitario, por lo que definimos un espacio de un cubo de 2 unidades de largo, como se muestra en la figura a continuación.

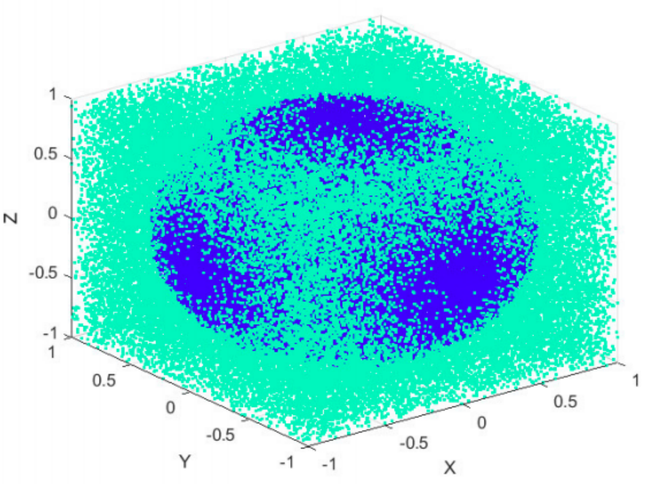

Figura 4. Representación gráfica del método Monte Carlo para calcular el volumen de una esfera con 1 millón de puntos

En la imagen, los puntos azul fuerte forman la esfera, mientras que los puntos aqua son todos aquellos que caen fuera de la figura, en el cubo cuyo volumen es conocido.

A. Algoritmo

De manera similar a las integrales de la función gamma, para iniciar el método se deben obtener números aleatorios del tamaño de puntos deseados (points). Se realiza una matriz del número de dimensiones (n) deseadas por el número de puntos usando el comando rand(n,points). Para evaluar si los puntos caen dentro de la esfera, deben cumplir con una restricción basada en la ecuación de una esfera en tres dimensiones: $x^2 + y^2 + z^2 \leq r^2$. Esta restricción es completamente escalable a esferas de todas las dimensiones, elevando el número de cordenadas al cuadrado y que la suma de estos sea menor o igual a uno, ya que se está trbajando con una esfera unitaria.

Si imaginamos la esfera como si estuviera circunscrita en un cubo con dos unidades de largo (de -1 a 1), la razón de números que cumplen estar dentro de la esfera entre la cantidad total de puntos es igual al volumen de la esfera sobre el volumen total del cubo. Para cada dimensión, el volumen de la cubo (o hiper espacio) es igual a 2^n.De esta forma es posible aproximar el volumen de la esfera en cualquier dimensión.

B. Resultados

Los resultados más exactos que se pudieron obtener, con 10,000,000 puntos, se despliegan en la tabla siguiente.

Dimensiones	Analítico	Monte-Carlo
1	2	2
2	3.1416	3.1416
3	4.1821	4.1905
4	4.9451	4.9344
5	5.2646	5.2629

En la Figura 5 se muestra graficado porcentaje de error en función del número de dimensiones de la hiperesfera. Se puede observar que la variabilidad del error incrementa conforme aumenta el número de dimensiones. Otra conclusión importante es que el error disminuye significativamente conforme mayor sea la cantidad de puntos que se utilizan para hacer la integral. En 11 dimensiones el error baja de 5.649 % a 0.916 % cuando se pasa de usar 10^6 a 10^7 puntos, respectivamente.

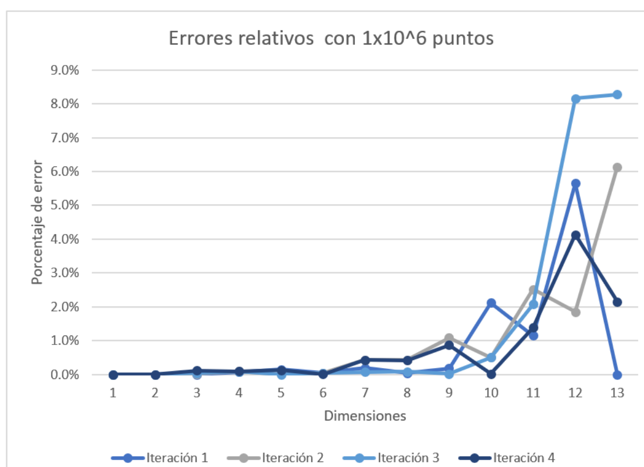

Figura 5. Representación gráfica de los errores relativos

V. CONCLUSIONES Y LIMITACIONES

El método de Monte Carlo representa un tipo de pensamiento probabilistico, distinto al determinista, con el cual los estudiantes están más acostumbrados. Trabajar con métodos estadísitcos y poder manipular esta técnica para encontrar respuestas de una manera más sencilla representa una herramienta muy útil y poderosa, además de brindar un acercamiento distinto a la resolución de problemas.

Como parte de este proyecto, se evaluó la función gamma con el método de Monte Carlo. A primera instancia, los errores fueron muy grandes y tenían altos porcentajes de error. Se logró incrementar la exactitud haciendo varias iteraciones (5,000) con la misma cantdad de puntos (100,000) para subsequentemente sacar un promedio de los datos. Los resultados obtenidos fueron bastante satisfactorios, pero al insertarlos en la Bessel, fue evidente que no eran lo suficientemente exactos. Esto porque la función gamma se utiliza en cada iteración del cálculo de cada función Bessel. Se concluye que para funciones muy delicadas, la magnitud del error escala de manera impresionante. La comparación visual permite ver claramente la diferencia entre la Bessel usando Monte Carlo con la función de Matlab. Como consecuencia, es evidente que el método de Monte Carlo para funciones de dos dimensiones puede ser fácilmente remplazado con algoritmos extremadamente sencillos de Matlab como son *sum()* o *trapz()*.

Sin embargo, al trabajar en mayores dimensiones, el método adquiere gran revelancia, incluso teniendo en cuenta el rango de error. Esto se de debe a que calcular integrales de más de tres dimensiones con cualquier otro método visto en clase sería extremadamente complicado y gastaría muchos recursos operativos. Igualmente, es relevante comentar que, a medida que la dimensión aumenta, el error y el tiempo de ejecución del programa también lo hace. El incremento en el tiempo se puede justificar dada la cantidad de datos que se deben de procesar. Asimismo, es importante notar que el error no se dispara al incrementar el número de dimensiones como lo hacen algunos otros métodos, pero tampoco se puede reducir el porcentaje de error al aumentar los puntos indiscriminadamente, como sucede con el step size en otros métodos. Los errores pequeños se mantienen dentro de un rango que es practicamente imposible disminuir, y los errores en dimensiones grandes incrementan por algún factor todavía desconocido.

A. Trabajo a futuro

El componente central del método de Monte Carlo es la generación de puntos aleatorios (experimentos), lo que nos lleva a la pregunta: ¿cómo generamos números verdaderamente aleatorios? En el caso de Matlab, podemos utilizar la función $rand$. Pero, este algoritmo en realidad es pseudoaleatorio, ya que después de cierto número de iteraciones nos generaría los mismos números, en el mismo orden. Existen alternativas para generar números verdaderamente aleatorios. Dos ejemplos son: usar un proceso físico (como sacar números en un sorteo) generando una tabla de números aleatorios, y utilizar un generado de ruido. Para mejorar los resultados de Monte Carlo sería relavante considerar estas dos opciones, aunque utilizar un algoritmo generador de números pseudoaleatorios sigue siendo la opción más eficiente, computacionalmente hablando.

Como vimos en el calculo de los volúmenes de las esferas, para dimensiones muy grandes ($n \geq 10$) el error crece mucho, lo que se puede deber a que el volumen de la esfera es cada vez menor y es menos probable que los puntos caigan dentro del volumen deseado. La idea para tratar de solucionar este problema fue aumentar el radio de la n-esfera. Sin embargo, nos dimos cuenta que esto no funcionaba, ya que el espacio a considerar también se incrementa y tenemos el mismo problema. Una opción que podría funcionar, sería considerar una esfera de mayor radio que la de interés (como nuestro espacio total) con volumen conocido. De esta forma, se podria reducir el error de manera considerable. En conclusión, basta con encontrar un volumen de mismas dimensiones que acote completamente el volumen que se intenta buscar. Creemos que mientras menos espacio se "desperdicie", más acertado se volverá el algoritmo de Monte Carlo, reduciendo el error relativo considerablemente.

REFERENCIAS

[1] J. I. Illana, Departamento de Física Teórica y del Cosmos: Universidad de Granada , 52 (2013).

[2] S. D. Poisson, *Recherches Sur La Probabilité Des Jugements En Matière Criminelle et En Matière Civile Precédées Des Règles Générales Du Calcul Des Probabilités Par Sd Poisson* (Bachelier, 1837).

[3] G. Arfken and H. Webber, *Mathematical Methods for Physicists* (Academic Press, 2000).

[4] J. V. Guttag, *Introduction to Computation and Programming Using Python* (The MIT Press, 2016).

[5] S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers* (McGraw-Hill, 2011).

[6] N. T. Thomopoulos, *Essentials of Monte Carlo Simulation: Statistical Methods for Building Simulation Models* (Springer, 2012).

VI. APÉNDICE: CÓDIGO

```
%% Proyecto Final - Monte Carla

clear all;
close all;
clc

fprintf("=== PROYECTO FINAL - Equipo Quintessence ===\n \n")
fprintf("** ENTER cuando se soliciten datos para usar valores default \n")
fprintf("** Sugerencias de los rangos para los datos en [CORCHETES] \n \n")
fprintf(" 1) C lculo de funci n Gamma y su error \n 2) Mejora del c lculo con histogramas \n" ...
    + " 3) Comparaci n de funciones Bessel \n 4)  rea de un c rculo (2 dimensiones) \n" ...
    + " 5) Volumen de una esfera (3 dimensiones) \n 6) C lculo del volumen de una hiperesfera (n dimensiones) \n \n");

n = input('Escoja una opci n: ');
switch n

    case -1
        fprintf('Programa terminado. Gracias :)');

    case 1
        fprintf('\n== C lculo de funci n Gamma y su error ==\n')

        z = input('Ingrese argumento de Gamma [valores mayores a 0]: ');
        if isempty(z); z = 4; disp("4"); end

        n = input('Ingrese n mero de iteraciones [valores positivos menores a 10^7]: ');
        if isempty(n); n = 10^7; disp("10^7"); end
        fprintf('\n');

        GammaMC(z,n,1);

    case 2
        fprintf('\n== Mejora del c lculo con histogramas ==\n')

        z = input('Ingrese argumento de Gamma [valores mayores a 0]: ');
        if isempty(z); z = 4; disp("4"); end

        n = input('Ingrese n mero de experimentos [valores positivos de 10^4 a 10^6]: ');
        if isempty(n); n = 10^4; disp("10^4"); end

        k = input('Ingrese n mero de veces que se realizar  Monte Carlo [valores positivos de 10^3 a 10^5]: ');
        if isempty(k); k = 10^3; disp("10^3"); end
        fprintf('\n');

        Histograma(z,n,k);

    case 3
        fprintf('\n== Comparaci n de funciones Bessel ==\n')
        v = input('Ingrese Bessel a calcular [valores enteros no negativos]: ');
        if isempty(v); v = 1; disp("J_1"); end
        fprintf('\n');

        BesselMC(v);

    case 4
        fprintf('\n==  rea de un c rculo (2 dimensiones) ==\n')
        n = input('Ingrese n mero de experimentos [valores positivos menores a 10^4]: ');
        if isempty(n); n = 5*10^3; disp("5*10^3"); end

        Circle(n);

    case 5
        fprintf('\n== Volumen de una esfera (3 
```

```
            dimensiones)␣==\n')
        n = input('Ingrese␣n mero␣de␣experimentos␣[valores␣positivos␣menores␣a␣10^4]:␣');
        if isempty(n); n = 10^4; disp("10^4"); end

        Sphere(n)

    case 6
        fprintf('\n==␣Volumen␣de␣una␣hiperesfera␣(n␣dimensiones)␣==\n')
        n = input('Ingrese␣n mero␣de␣experimentos␣[valores␣positivos␣menores␣a␣10^8]:␣');
        if isempty(n); n = 10^7; disp("10^7"); end
        fprintf('\n');

        dim = input('Ingrese␣n mero␣de␣dimensiones␣[enteros␣positivos␣menores␣a␣18]:␣');
        if isempty(dim); dim = 5; disp("5"); end

        NSphere(n,dim)

    otherwise
        fprintf('El␣valor␣ingresado␣no␣es␣una␣opci n␣v lida.␣Favor␣de␣escoger␣otra␣opci n␣o␣ingresar␣"-1"␣para␣salir.')
end

fprintf('Favor␣de␣volver␣a␣correr␣el␣programa␣si␣se␣quiere␣probar␣otra␣parte␣del␣proyecto.')


%% 1) GammaMC, calcula los resultados de la gamma

function [AreaGamma] = GammaMC(z,n,flag)

    format long;

    if z<1 %condicional para gammas en el intervalo (0,1)
       x = linspace(0.0001,1,n);
    else
       x = linspace(0,1,n);
    end

    %funci n gamma
    G = ((1-x)./x).^(z-1).*exp(-((1-x)./x))./x.^2;
    %vectores aleatorios
    xp = rand(1,n); yp = rand(1,n)*max(G);
    ygamma = ((1-xp)./xp).^(z-1).*exp(-((1-xp)./xp))./xp.^2;

    INRANGE=yp<=ygamma; %aciertos

    Ratio=sum(INRANGE)/n; %raz n entre aciertos y total
    AreaSquare=x(end)*max(G); %volumen del espacio
    AreaGamma=AreaSquare*Ratio; %resultado de la integral
    error=abs((gamma(z)-AreaGamma)/gamma(z))*100;

    if(flag==1)
        disp("Valor real de gamma: " + gamma(z))
        disp("Valor de gamma con Monte Carlo: " + AreaGamma)
        disp("Error relativo: " + error + "%");
        disp(" ")
    end

end

%% 2) Histograma, obtiene el promedio de muchos c lculos con Monte Carlo

function Histograma(z,n,k)

    Results = zeros(1,k);
    figure(1)

    % el c digo comentado se utiliz  para guardar las animaciones

%     myVideo = VideoWriter('HistogramGif');
%     myVideo.FrameRate = 125;
%     open(myVideo)

    %ciclo para graficar histograma de resultados
    for i = 1:k
        AreaGamma = GammaMC(z,n,2);
        Results(i) = AreaGamma;
        h = histogram(Results);
%         pause(10^(-8));
%         frame = getframe(gcf); %get frame
```

```
%        writeVideo(myVideo, frame);
    end

    h.FaceColor = [0 0.7 0.5];
    ylabel('Frecuencias'); xlabel('Gamma(x)')
    axis([gamma(z)-0.4,gamma(z)+0.4,0,k*.2]); grid on;

%      close(myVideo)
    disp("Funci n Gamma:"+string(gamma(z)))
    %obtiene el promedio de los resultados
    disp("Promedio de histograma:"+string(mean(Results)))

end

%% 3) Comparaci n de funciones Bessel, con y sin Monte Carlo

function BesselMC(v)

    syms x
    load('GammaN.mat');
    %estos datos resultan de llamar la funci n anterior, GammaMC,
    %y posteriormente limpiar los datos con la funci n posterior, Histogamma;
    %sin embargo el proceso tarda en ejecutarse, por lo que ya dejamos
    %los datos listos para utilizarse en esta funci n.

    J=0;
    %ciclo que calcula la funci n Bessel a partir de la Gamma con Monte Carlo
    for m=0:199-(v)
         J = J + (-1)^m/(factorial(m)*GammaN(m+v+1)).*(x/2).^(2*m+v);
    end

    %se grafica la Bessel de Matlab y la nuestra, para compararlas
    fplot(besselj(v, x),'LineWidth',1.2);
    hold on; grid on;
    axis([-12,12,-0.7,1.1]);
    fplot(J,'--','LineWidth',2.5);
    ylabel('J_v(x)'); xlabel('x');
    legend1="J_"+string(v);
    legend2="J_"+string(v)+" (Monte Carlo)";
    title1="Bessel functions of the first kind (J_"+string(v)+")";
    legend(legend1,legend2,'Location','Best');
    title(title1);

end

%% 4)  rea  de un c rculo, c lculo con Monte Carlo (2 dimensiones)

function Circle(n)

    x=linspace(-1,1,n);
    figure(1); %plot c rculo
    plot(x,sqrt(1-x.^2),'b',x,-sqrt(1-x.^2),'Color',1/255.*[253,98,94])
    hold on


    %vectores aleatorios
    xp = rand(1,n)*2-1;
    yp = rand(1,n)*2-1;

    NOTINRANGE = zeros(1,n); %fallos
    INRANGE = zeros(1,n); %aciertos

    for i = 1:n
        if xp(i)^2+yp(i)^2 <=1
            INRANGE(i) = 1;
            plot(xp(i),yp(i),'.','Color',[.24,.92,.73])
        else
            NOTINRANGE(i) = 1;
            plot(xp(i),yp(i),'.','Color',[0,0,1])
        end
    end

    Ratio=sum(INRANGE)./n;
    AreaSphere=Ratio*4;
    error=abs((pi-AreaSphere)/pi)*100;

    % Display de resultados
    disp(" ");
    disp("N mero de experimentos: "+string(n));
    disp(" rea  del c rculo: "+string(pi));
    disp(" rea  del c rculo con Monte Carlo: "+string(AreaSphere));
    disp("Error relativo: " + error + "%");
    disp(" ");
```

```
end

%% 5) Volumen de una esfera, c lculo con Monte Carlo (3 dimensiones)

function Sphere(n)

    % vectores aleatorios
    xp = rand(1,n)*2-1;
    yp = rand(1,n)*2-1;
    zp = rand(1,n)*2-1;

    % plot Esfera en 3D
    [X,Y,Z] = sphere(100);
    figure(2); surf(X,Y,Z); alpha 0.05;
    shading interp; hold on

    NOTINRANGE = zeros(1,n); % fallos
    INRANGE = zeros(1,n); %aciertos

    % el c digo comentado se utiliz  para guardar las animaciones

%     myVideo = VideoWriter('SphereGif'); %open video file
%     myVideo.FrameRate = 200;  %can adjust this, 5 - 10 works well for me
%     open(myVideo)

    %gr fica de "dardos"
    for i = 1:n
        if xp(i)^2+yp(i)^2+zp(i)^2 <=1
            INRANGE(i) = 1;
            plot3(xp(i),yp(i),zp(i),'.','Color',[0,0,1])
        else
            NOTINRANGE(i) = 1;
            plot3(xp(i),yp(i),zp(i),'.','Color',[.24,.92,.73])
        end
%         frame = getframe(gcf); %get frame
%         writeVideo(myVideo, frame);
%         pause(10^(-8))
    end

%     close(myVideo)

    Ratio = sum(INRANGE)./n;
    VolSphere = Ratio*8;
    error=abs((pi*4/3-VolSphere)/(pi*4/3))*100;

    % Display
    disp(" ");
    disp("N mero de experimentos: "+string(n));
    disp("Volumen de la esfera: "+string(pi*4/3));
    disp("Volumen de la esfera con Monte Carlo: "+string(VolSphere));
    disp("Error relativo: " + error + "%");
    disp(" ");

end

%% 6) Volumen de una hiperesfera, c lculo con Monte Carlo (n dimensiones)

function NSphere(n,dim)

    points = rand(dim,n).*2-1;
    INRANGE = zeros(1,n);

    for i = 1:n
        if sum(points(:,i).^2)<=1
            INRANGE(i) = 1;
        end
    end

    Ratio= sum(INRANGE)./n;
    VolSphere = Ratio*2^dim;
    VolAnalitico = pi^(dim/2)/gamma(dim/2+1);
    error=abs((VolAnalitico-VolSphere)/(VolAnalitico))*100;

    % Display
    disp(" ");
    disp("N mero de experimentos: "+string(n))
    disp(" ");
    disp("Volumen de una esfera de "+string(dim)+" dimensiones")
    disp("Volumen anal tico: "+string(VolAnalitico))
    disp("Monte Carla: "+string(VolSphere))
    disp("Error relativo: " + error + "%");
    disp(" ");

end
```