POLINOMIOS

Los polinomios de hermite a través de recurrencias integrales

Hermite polynomials via integral recurrence relations

Miguel Ángel Hurtado Benavides
mangelhb@unimonserrate.edu.co
Fundación Universitaria Monserrate – Unimonserrate

Resumen

Los polinomios de Hermite son importantes en algunos campos de las matemáticas, como probabilidad, teoría de error y ecuaciones diferenciales; ciencias, física cuántica; química; e ingeniería, como la computación y aplicaciones de las anteriores mencionadas. El objetivo de este escrito es presentar una forma novedosa de obtener los polinomios de Hermite, mediante recurrencias integrales y mostrar una aplicación a la computación científica a través de su codificación, para llegar a objetos matemáticos relacionados como la integral de Gauss, la distribución de probabilidad normal y la función error.

Palabras clave: Polinomios de Hermite; recurrencia integral.

Abstract

Hermite polynomials are of fundamental importance in many fields of mathematics, such as probability, error theory and differential equations; in science, such as quantum physics and chemistry; and in engineering, such as computing and applications of the above mentioned. Therefore, the study of this family of polynomials is always relevant. The objective of this paper is to present a novel way of obtaining Hermite polynomials by means of integral recurrences and to show an application to scientific computing through the codification of said recurrences, with which mathematical objects related to such polynomials such as the Gauss integral, the normal probability distribution and the error function are reached.

Keywords: Hermite polynomials; integral recurrence.

Introducción

Academia (2023) menciona que los polinomios de Hermite fueron definidos por Pier Simón en 1810 y estudiados en 1859 por Pafnuty Chebyshev. Al parecer el trabajo de Chebyshev pasó desapercibido, pues más tarde recibieron el nombre de Charles Hermite, quien los describió en su forma multidimensional en 1865. Los polinomios de Hermite son importantes por sus variadas aplicaciones en probabilidad, como la obtención de los valores de la distribución normal; en física, en las funciones de onda de un oscilador cuántico (Álvarez, 2021); en la teoría de grafos, en la combinatoria de un gráfico simple (Patarrollo, 2019). Además, estos polinomios tienen aplicaciones en computación científica, como se muestra en los lenguajes de programación de Python (2022); Wolfram (2023) y Matlab (2023), donde se encuentran alojadas librerías con programas basados en las propiedades matemáticas de estos polinomios, proporcionando métodos numéricos estándar para la solución de integrales indeterminadas o como la que expone Saenz (2019) la cuadratura de Gauss.

Los polinomios de Hermite aplicados a la probabilidad (Patarroyo, 2019) aparecen definidos a continuación:

Definición 1

Los polinomios de Hermite es una sucesión $\{He_m(x)\}_{m\geq 0}$ que se definen en el intervalo $-\infty < x < \infty$, mediante la serie generadora:

$$He(x,t) = e^{xt-\frac{t^2}{2}} = \sum_{m=0}^{\infty} He_m(x)\frac{t^m}{m!},$$

Función de distribución de probabilidad normal estándar de Gauss

La función de densidad de probabilidad normal estándar de Gauss, $\varphi(t)$, surge de forma natural de la serie generadora $He_m(x,t)$, evaluada para $x = 0$ y multiplicando por $1/\sqrt{2\pi}$, esto es:

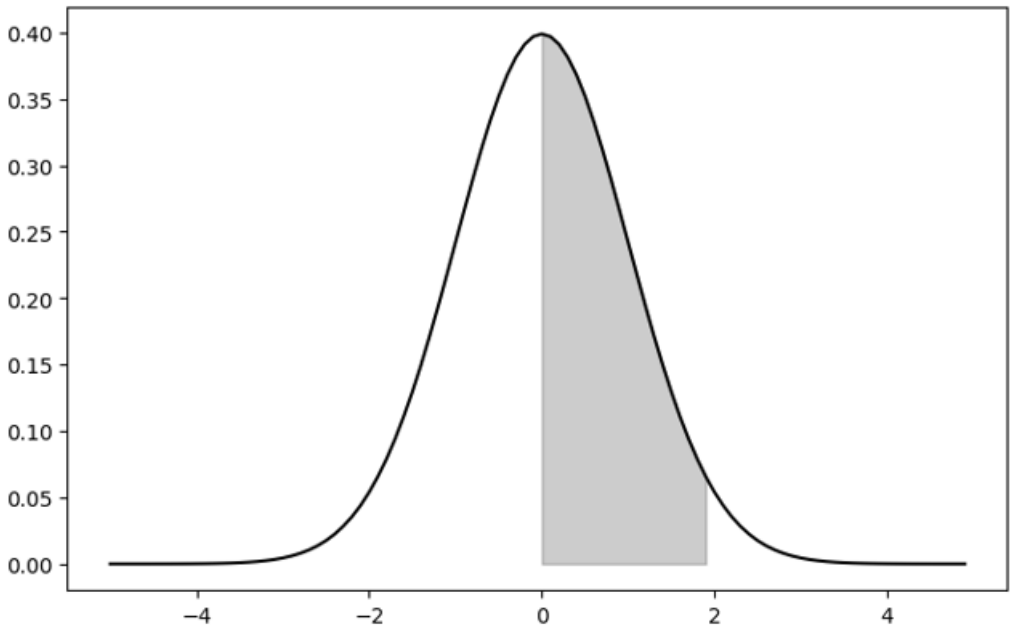
$$\varphi(t) = \frac{1}{\sqrt{2\pi}}e^{-\frac{t^2}{2}} = \frac{1}{\sqrt{2\pi}}\sum_{m=0}^{\infty} He_m(0)\frac{t^m}{m!}.$$

Ahora, integrando en el intervalo de $[0, z]$ con respecto a la variable t, se llega a la función de distribución de probabilidad normal estándar de Gauss, esto es:

$$P[0 \leq Z \leq z] = \frac{1}{\sqrt{2\pi}}\int_0^z e^{-\frac{t^2}{2}}\,dt$$

Una representación gráfica de esta integral es el área bajo la curva de la función $\varphi(t)$, en un intervalo dado de $[0,z]$, cuyo valor es la probabilidad de que ocurra un evento con una distribución normal estándar, lo cual se escribe como $P[0\leq Z\leq z]$. En la siguiente figura se ejemplifica $P[0\leq Z\leq 2]$.

Figura 1. Representación gráfica de $P[0\leq Z\leq 2]$



Fuente: elaboración propia.

Por otro lado, los polinomios de Hermite aplicados a la física aparecen definidos en Romano y Rota (1978) de la siguiente manera:

Definición 2

Los polinomios de Hermite $\{H_m(x)\}_{m\geq 0}$ se definen en el intervalo $-\infty< x <\infty$,mediante la serie generadora:

$$H(x,t)=\sum_{m=0}^{\infty} H_m(x)\frac{t^m}{m!}=e^{2xt-t^2},$$

En particular, para $x=0$, esta serie se relaciona con la función error de Gauss $erf(z)$.

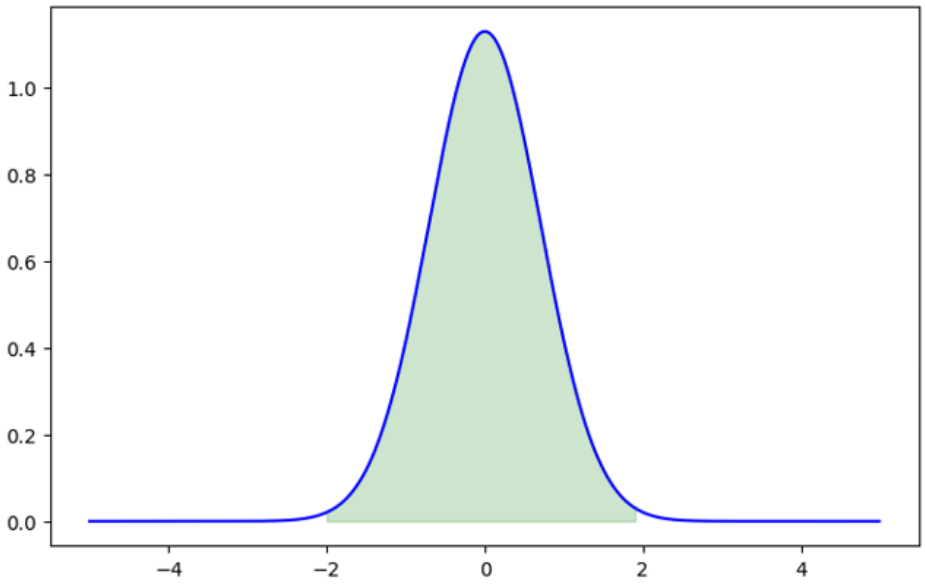
Función error de Gauss

La función error de Gauss "$erf(z)$" se define como la integral definida (Cruz y Tetlalmatzi, 2015),

$$erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, dt.$$

Una representación gráfica de los valores numéricos de $erf(z)$ están dados por el área bajo la curva de tipo campana de Gauss como se presenta a continuación:

Figura 2. Representación gráfica de $erf(2)$



Fuente: elaboración propia.

No existe método analítico para solucionar las integrales definidas de la función de distribución de probabilidad y de la función error de Gauss. Por eso, en la literatura existen métodos numéricos para dar valores de aproximación, por ejemplo, por medio de polinomios de MacLaurin o de Taylor para la función exponencial.

En Cruz y Tetlalmatzi (2015) se demuestra el caso particular de la integral impropia, llamada integral de Gauss, que tiene el siguiente resultado:

$$\int_0^\infty e^{-t^2}\, dt = \frac{\sqrt{\pi}}{2}$$

Transponiendo términos se tiene que,

$$\frac{2}{\sqrt{\pi}}\int_0^{\infty} e^{-t^2}\,dt = \frac{1}{\sqrt{\pi}}\int_{-\infty}^{\infty} e^{-t^2}\,dt = 1$$

Con lo cual se tiene un valor importante del área bajo toda la curva de la función error.

Por otro lado, para obtener valores aproximados de esta integral en un intervalo definido, es decir, valores de la función $erf(z)$, y de los valores de la función de distribución estándar, es a través de truncamientos de las series generadoras $H(0,t)$,$He(0,t)$ respectivamente. Para esto es necesario obtener los polinomios de Hermite, que pueden generarse desde diversos métodos, según la literatura. En este escrito se opta por un método a través de recurrencias integrales, que se deduce y se demuestra en Hurtado (2020):

Primera recurrencia integral de polinomios de Hermite

La sucesión $\{He_m(x)\}_{m\geq 0}$ de polinomios de Hermite para aplicaciones en probabilidad satisface la recurrencia integral:

$$He_m(x) = m\int_0^x He_{m-1}(t)\,dt - He'_{m-1}(0), para\ todo\ m \geq 0\ con\ He_0(x) = 1$$

Segunda recurrencia integral de polinomios de Hermite

La sucesión $\{H_m(x)\}_{m\geq 0}$, de polinomios de Hermite para aplicaciones en física satisface la recurrencia integral:

$$H_m(x) = 2m\int_0^x H_{m-1}(t)\,dt - H'_{m-1}(0), para\ todo\ m \geq 0\ con\ H_0(x) = 1.$$

Por ejemplo, mediante estas formulas de recurrencias integrales se tienen los primeros polinomios de Hermite correspondientemente:

$$He_0(x) = 1, He_1(x) = x, He_2(x) = x^2 - 1, He_3(x) = x^3 - 3x,$$

$$He_4(x) = x^4 - 6x^2 + 3, He_5(x) = x^5 - 10x^3 + 15x, \ldots$$

y

$$H_0(x) = 1, H_1(x) = 2x, H_2(x) = 4x^2 - 2, H_3(x) = 8x^3 - 12x$$

$$H_4(x) = 16x^4 - 48x^2 + 12, H_5(x) = 32x^5 - 160x^3 + 120x, \ldots$$

En este artículo se aplican las anteriores recurrencias integrales, las cuales son novedosas en cuanto a la forma de obtener los polinomios de Hermite, este tipo de recurrencias integrales también se demuestran en Carrillo y Hurtado (2021) a través de los funcionales que definen las series generadoras de las sucesiones de Appell. En particular, la sucesión $\{He_m(x)\}_{m\geq 0}$ conforma una sucesión de Appell, esto significa que:

$$\frac{d}{dx}He_m(x) = mHe_{m-1}(x).$$

Esto se puede verificar de forma inmediata derivando la primera recurrencia integral de polinomios de Hermite.

Justificación

Para Olvera, Rodríguez, González y Gutiérrez (2014), la computación en la ciencia es la disciplina que estudia el manejo y la aplicación de las disciplinas en que está basada las matemáticas y las ingenierías, así como el estudio de productos y procedimientos, como los algoritmos basados en fórmulas y propiedades matemáticas. Por su parte, Gallo (2023) destaca que parte de la evolución de las computadoras se dio gracias al ingenio y creatividad de matemáticos que aplicaron sus conocimientos a la computación; por otro lado, Calvo y Calvo (2020) mencionan la importancia de llevar a las aulas el estudio de la mecánica cuántica. Este panorama justifica este trabajo, en particular sobre las aplicaciones en computación científica que pueden tener las aludidas recurrencias integrales que generan los polinomios de Hermite.

Objetivo

Cada vez que aparece un nuevo método, algoritmo o fórmula para manipular un objeto matemático, surge el cuestionamiento sobre la naturaleza del procedimiento respecto a su aplicabilidad en las ciencias e ingeniería, en la operatividad y sus propiedades, en su comprensión en cuanto lo cognitivo, en la facilidad de su codificación en programación; todo esto para sacar el mayor provecho de sus bondades encontradas. A partir de la bibliografía consultada, surge la pregunta problema: ¿Cómo medir la convergencia de los objetos relacionados con los polinomios de Hermite a través de recurrencias integrales?

Para dar respuesta a la pregunta problema, en este trabajo se propone como objetivo: “medir la convergencia de los objetos relacionados con los polinomios de Hermite a través de recurrencias integrales”. El documento está dividido en tres secciones: metodología, resultados, discusión y conclusiones.

Metodología

Este trabajo es una investigación aplicada sobre la investigación realizada por Hurtado (2020). Según Ortega (2023), la investigación aplicada tiene como finalidad aplicar conocimientos científicos hallados en una investigación básica. En particular, este escrito está enfocado a la aplicación en la computación científica del método para obtener los polinomios de Hermite a través de recurrencias integrales, avanzando en el conocimiento de las propiedades y aplicaciones de estos.

Un equipo idóneo para evidenciar la aplicabilidad de las recurrencias integrales de polinomios de Hermite en la computación científica es un software de lenguaje de programación. Para esta investigación se opta por usar el lenguaje de programación de Python 3.0, alojado en cuadernos de Google Colaboratory (2020), pues en este se encuentran programados objetos que arrojan los valores numéricos de $erf(z)$, y de $\mathrm{P}[0\leq Z \leq z]$, con lo cual se pueden hacer de forma inmediata comparaciones gráficas y análisis numérico y cota de error.

Mediante comando de Python statistics (2022), se obtienen algunos de los valores de la función distribución de probabilidad normal estándar, a saber $P[0 \leq Z \leq 9]$. El programa y los mencionados valores se encuentran en la siguiente figura:

Figura 3. Comando de Python que arroja los valores $P[0 \leq Z \leq 9]$

```
import scipy.stats as stats
from scipy.stats import norm
z = -1
while z < 9:
  z += 1
  print("P(0 < Z <", z,") =",stats.norm.cdf(z)-0.5)
```

```
P(0 < Z < 0 ) = 0.0
P(0 < Z < 1 ) = 0.3413447460685429
P(0 < Z < 2 ) = 0.4772498680518208
P(0 < Z < 3 ) = 0.4986501019683699
P(0 < Z < 4 ) = 0.4999683287581669
P(0 < Z < 5 ) = 0.4999997133484281
P(0 < Z < 6 ) = 0.4999999990134123
P(0 < Z < 7 ) = 0.49999999999872013
P(0 < Z < 8 ) = 0.49999999999999933
P(0 < Z < 9 ) = 0.5
```

Fuente: elaboración propia.

Para valores $Z \geq 9$, el programa redondea a 0.5, es decir, $P[0 \leq Z \leq \infty] = 0.5$.

Por medio del comando de Python math.*erf*() Method (2022), se obtienen algunos valores de $erf(z)$, como se muestra en la siguiente figura:

Figura 4. Comando de Python que arroja valores de $erf(z)$

```
1  import math
2  z = -1
3  while z < 6:
4    z += 1
5    print("erf(",z,") =",math.erf(z))
```

```
erf( 0 ) = 0.0
erf( 1 ) = 0.8427007929497149
erf( 2 ) = 0.9953222650189527
erf( 3 ) = 0.9999779095030014
erf( 4 ) = 0.9999999845827421
erf( 5 ) = 0.9999999999984626
erf( 6 ) = 1.0
```

Fuente: elaboración propia.

Las hipótesis que se tienen para esta investigación son las siguientes:

> Hn: Los objetos relacionados con los polinomios de Hermite no se pueden aproximar a través de recurrencias integrales.
>
> Ha: Los objetos relacionados con los polinomios de Hermite se pueden aproximar a través de recurrencias integrales.

Para aceptar o rechazar la hipótesis nula, se propone aplicar las recurrencias integrales de polinomios de Hermite a través de la construcción de un instrumento de investigación, que consiste en la programación del algoritmo de recurrencia integral de estos y los objetos matemáticos derivados, para luego implementarlo y recoger los datos simbólicos, gráficos y/o numéricos que arroje. La estrategia es: analizar las aproximaciones de las representaciones gráficas, luego, comparar las aproximaciones numéricas que arroja el instrumento, con respecto a los valores expuestos en las figuras 3 y 4; y por último analizar la complejidad del algoritmo de recurrencia.

El instrumento de investigación debe arrojar los primeros h polinomios de Hermite a través de las recurrencias integrales, según sea el caso de aplicación. A partir de estos, se dan aproximaciones o truncamientos a las funciones $erf(z)$, $\varphi(t)$, y $P[0 \leq Z \leq z]$, esto es:

$$\varphi(t) \approx \frac{1}{\sqrt{2\pi}} \sum_{m=0}^{h} He_m(0) \frac{t^m}{m!}, H(0,t) \approx \sum_{m=0}^{h} H_m(0) \frac{t^m}{m!},$$

$$P[-z \leq Z \leq z] \approx \frac{1}{\sqrt{2\pi}} \int_{-z}^{z} \sum_{m=0}^{h} He_m(0) \frac{t^m}{m!} dt, erf(z) \approx \frac{2}{\sqrt{\pi}} \int_{0}^{z} \sum_{m=0}^{h} H_m(0) \frac{t^m}{m!} dt,$$

Como los valores que arroja el comando de Python es $P[0 \leq Z \leq 9]$, se debe buscar un valor h que se aproxime a la gráfica de $\varphi(9)$.

Después de obtener dicha aproximación gráfica, se propone realizar primero una búsqueda de un valor menor a $\alpha = 1.0 \text{ X } 10^{-15}$ como indicador de prueba de precisión, entre los valores que arroje la función programada, con respecto, a los valores que arrojen el comando alojado en las librerías de Python. Una forma de obtener valores α_h menores al indicador de prueba α, es mediante la prueba del error relativo, que consiste en la comparación del valor relativamente exacto, con respecto a los valores encontrados por el objeto de prueba, la fórmula del error relativo es la que sigue:

$$error_{relativo} = \frac{|X - X_h|}{X},$$

Donde X es el valor relativamente exacto y, X_h son los sucesivos valores de prueba, con lo cual se busca un indicador de calidad en la medida (Chapra y Canale. 2007).

Según el análisis de algoritmos de Duch (2007), un código recurrente de derivadas o integrales de funciones polinómicas es de complejidad $O(h)$, donde h es el grado del polinomio, entonces, como indicador se tiene que el algoritmo recurrente del instrumento no sea mayor.

Luego de obtener un valor del error relativo menor a α, se procede a mantener alojados los polinomios de aproximación a $P[0 \leq Z \leq z]$ y de $erf(z)$ en dos variables correspondientes a cada uno de ellos.

Resultados

Como primer resultado es la aplicación de dichas recurrencias integrales para la construcción del instrumento de investigación; un segundo resultado es el análisis de la complejidad del mencionado algoritmo recurrente; y como tercer resultado, se implementa el instrumento de investigación para hacer un análisis numérico sobre la aproximación de objetos relacionados y construidos con los polinomios de Hermite. En la siguiente figura se ilustra el código de programación base para la investigación que se propone en este trabajo.

Figura 5. Instrumento de investigación, basado en código iterativo de las recurrencias integrales de polinomios Hermite

```
1  from sympy import *                    # Libreria
2  # Símbolos
3  t,x,y,He,He0,H,H0,PHe0,erf = symbols("t x y Hem(x) Hem(0) Hm(x) Hm(0) PHe(x) erf(x)")
4  def Hm(tipo,h):                        # Algoritmo de recurrencia integral
5    m = 0
6    H = 1
7    fgxt = 1
8    fac = 1
9    while m < h:
10       m = m + 1
11       dH = diff(H)                     # H'm(x)
12       dH0 = - dH.subs(x,0)             # H'm(0)
13       H = tipo*m*integrate(H, x) + dH0 # Recurrencia Integral
14       fac = fac*m                      # factorial
15       fgxt = fgxt + H*t**m/fac         # Función generadora de Hm(x)
16
17   fg0t = fgxt.subs(x,0)                # Función generadora de Hm(0)
18   densidad = tipo*fg0t/sqrt((3-tipo)*pi) # Función de densidad
19   distribucion = integrate(densidad,t)   # Función de distribución
20   return H,fgxt,fg0t,densidad,distribucion
21
22 def Grafica(F,G,w,a,b):
23   p = plot(F,(w,a,b),legend = False,show=False)
24   p.extend(plot(G,(w,a,b),line_color='r',show=False))
25   p.show()
```

Fuente: elaboración propia.

En la anterior figura, se encuentran los códigos de programación de dos funciones: la primera es una función realizada en lenguaje de programación simbólico de Sympy (2020), donde se codificó las recurrencias integrales de polinomios de Hermite y se programaron las funciones polinómicas dadas por la suma de los primeros términos de las series generadoras $He(x,t)$ y $H(x,t)$ con el propósito de obtener aproximaciones numéricas de las funciones $erf(z)$, $\varphi(t)$, y $P[0 \leq Z \leq z]$. El segundo código es una función que arroja la representación gráfica de los objetos matemáticos en mención, con lo cual se realizan las primeras aproximaciones a través de la observación del truncamiento de las series $He(x,t)$ y $H(x,t)$.

La función Hm toma dos parámetros “tipo” y “h” y devuelve los polinomios de Hermite que requiera el usuario. El primer parámetro indica el tipo de estos polinomios, es decir, con el número 1 se obtienen los polinomios He y con el número 2 los polinomios H; el segundo parámetro corresponde al h-ésimo polinomio que se requiere imprimir. En las líneas 5, 6 y 7 se encuentran las semillas o primeros valores “m” de los polinomios de Hermite “H” y de sus series generadoras “fgxt”. De la línea 8 a la 13, se encuentran codificados los algoritmos de las recurrencias integrales para obtener los primeros polinomios de Hermite y los primeros términos de sus series generadoras. En la línea 11 se da el comando para derivar los polinomios, en la fila 12 se evalúa dicha deri-

vada para $x= 0$, en la línea 13 se tiene codificadas las fórmulas de recurrencia de los polinomios de Hermite, en la línea 15 la suma de los primeros términos de la función generadora de los mismos polinomios. En esta misma línea 15 se tiene los valores de la anterior fila evaluados en $x= 0$, en la línea 17 se aloja el truncamiento de la función $H(0,t)$, en la línea 18 se guarda la aproximación a la función de densidad o campana de Gauss; en la línea 19 se aloja la aproximación de la función de distribución de probabilidad o la función error, según el requerimiento. Esta función devuelve en un vector los mencionados objetos matemáticos relacionados con estos polinomios, la posición [0] arroja los polinomios de Hermite requeridos, a saber: la posición [1] retorna la suma de los primeros términos de la serie generadora, [2] la misma suma evaluada en $x= 0$; con la que [3] se obtienen los polinomios que se aproximan a la función de densidad de probabilidad o para la función tipo campana de la función error; y [4] devuelve los polinomios que se aproximan a la función de probabilidad estándar $P[0 \leq Z \leq z]$ o para la función $erf(z)$, según el requerimiento.

Por otro lado, el programa Grafica (F, G, w, a, b), como su nombre lo indica, es el código que devuelve la representación gráfica de funciones “F” y “G”, alojados en la primera y segunda entrada, cuya variable independiente se aloja en la tercera entrada y, la cuarta y quinta entradas son para designar el intervalo del dominio de la función que se quiere visualizar.

Ejemplos de implementación de los programas Hm (tipo, h) y Grafica (F, G, w, a, b).

$$Hm(1,7)[0], \rightarrow, He_7(x) = x^7 - 21x^5 + 105x^3 - 105x,$$

$$Hm(2,8)[0], \rightarrow, H_8(x) = 256x^8 - 3584x^6 + 13440x^4 - 13440x^2 + 1680$$

Que son precisamente los polinomios de Hermite requeridos.

- Para obtener los primeros términos de las series generadoras

$$Hm(1,4)[1], \rightarrow, He_4(x,t) = 1 + xt + \frac{(x^2-1)}{2}t^2 + \frac{(x^3-3x)}{6}t^3 + \frac{(x^4-6x^2+3)}{24}t^4$$

$$Hm(2,3)[1], \rightarrow, H_3(x,t) = 1 + 2xt + \frac{4x^2-2}{2}t^2 + \frac{8x^3-12x}{6}t^3$$

- Para obtener primeros términos de las series $He(0,t)$ y $H(0,t)$

$$Hm(1,14)[2], \rightarrow, He_{14}(0,t) = 1 - \frac{t^2}{2} + \frac{t^4}{8} - \frac{t^6}{48} + \frac{t^8}{384} - \frac{t^{10}}{3840} + \frac{t^{12}}{46080} - \frac{t^{14}}{645120}$$

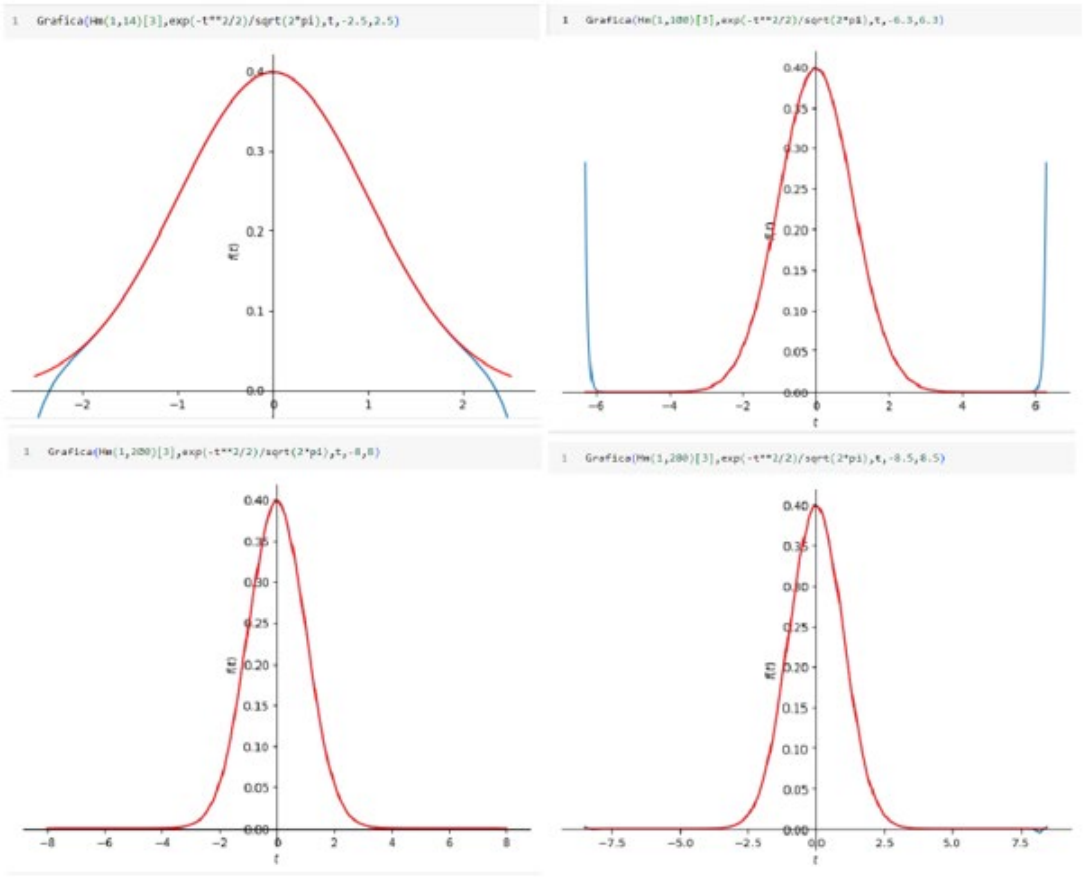
$$Hm(2,16)[2], \rightarrow, H_{16}(0,t) = 1 - \frac{t^2}{2} + \frac{t^4}{8} - \frac{t^6}{6} + \frac{t^8}{24} - \frac{t^{10}}{120} + \frac{t^{12}}{720} - \frac{t^{14}}{5040} + \frac{t^{16}}{40320}$$

Aproximación de las funciones $\varphi(t)$, $P[0 \leq Z \leq z]$ *y* $erf(z)$

En esta sección se muestra la implementación de los programas Hm y Gráfica para obtener aproximaciones sucesivas de la función de densidad o campana de Gauss de manera gráfica, y luego una aproximación numérica. Estos resultados se comparan respecto a los valores que arroja el comando de Python de la función de distribución de probabilidad normal estándar y el comando para de función error.

Ahora bien, haciendo uso de las recurrencias integrales a través del programa expuesto en la figura 5, se obtiene las siguientes aproximaciones gráficas de la función densidad de probabilidad normal estándar:

Figura 6. Aproximación gráfica de la función de densidad de probabilidad normal estándar



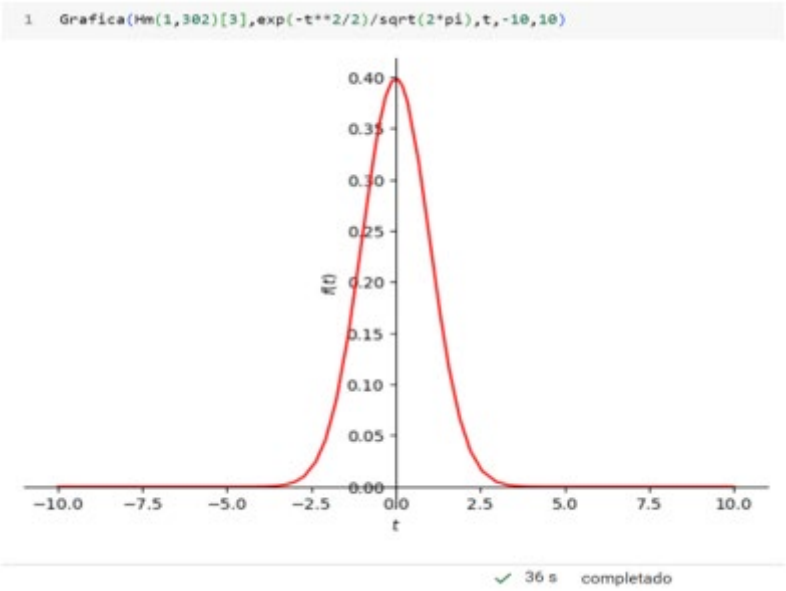
Fuente: elaboración propia.

En las cuatro gráficas, la línea roja es la representación gráfica de la función de densidad de probabilidad normal estándar, es decir, $\varphi(t) = 1/\sqrt{2\pi}\, e^{-t^{\wedge}2/2;}$ y la línea azul es la aproximación polinomial de la función en mención a través de las recurrencias integrales de polinomios de Hermite.

La primera gráfica (arriba izquierda) está dada por el polinomio de aproximación de grado 14; la segunda gráfica (arriba derecha) está dada por el polinomio de aproximación de grado 100. De estas dos primeras aproximaciones se observa que la curva azul no se aproxima a la curva roja para valores de $-6 \leq Z$ o $Z \geq 6$. Por tanto, estas aproximaciones aún no cumplen con el valor de determinación, es decir, $\alpha = 1.0\ X\ 10^{-15}$. La tercera gráfica (abajo izquierda) está dada por el polinomio de aproximación de grado 200, la cuarta gráfica (abajo derecha) está dada por el polinomio de aproximación de grado 280. Se observa que la curva azul se aproxima mucho más hasta el intervalo [-8,8]. Finalmente, se tiene una aproximación gráfica de $\varphi(t)$ en un intervalo que contiene al [-9,9] dado por el polinomio de aproximación de grado 302.

$$\varphi(t) \approx \frac{1}{\sqrt{2\pi}} \sum_{m=0}^{302} He_m(0) \frac{t^m}{m!}$$

Figura 7. Aproximación gráfica de la función de densidad de probabilidad normal estándar



Fuente: elaboración propia.

Sobre este último truncamiento, es posible hacer aproximaciones numéricas de la función de distribución de probabilidad normal estándar,

$$P[0 \leq Z \leq 9] \approx \frac{1}{\sqrt{2\pi}} \int_0^9 \sum_{m=0}^{302} He_m(0) \frac{t^m}{m!} dt$$

Haciendo uso del instrumento expuesto en la figura 5 se obtiene los siguientes valores:

Figura 8. Aproximaciones de valores $P[0 \leq Z \leq 9]$

```
1  ProbNormEst = Hm(1,302)[4]
2
3  z = -1
4  while z < 9:
5    z += 1
6    print("ProbNormEst(",z,") =",float(ProbNormEst.subs(t,z)))
```

```
ProbNormEst( 0 ) = 0.0
ProbNormEst( 1 ) = 0.3413447460685429
ProbNormEst( 2 ) = 0.4772498680518208
ProbNormEst( 3 ) = 0.4986501019683699
ProbNormEst( 4 ) = 0.4999683287581669
ProbNormEst( 5 ) = 0.49999971334842813
ProbNormEst( 6 ) = 0.49999999901341236
ProbNormEst( 7 ) = 0.4999999999987202
ProbNormEst( 8 ) = 0.4999999999999994
ProbNormEst( 9 ) = 0.5
```

✓ 36 s completado

Fuente: elaboración propia.

Los valores dados en la figura 8 son muy cercanos a los valores que se dan en la figura 3. Para verificar la calidad de la aproximación se deben comparar dichos valores. Haciendo uso de la fórmula de error relativo propuesto en la metodología, se tienen los siguientes valores α_h:

Tabla 1. Error relativo entre valores para $P[0 \leq Z \leq 9]$

h	X	Xh	Error relativo α_h
1	0,3413447460685429	0,3413447460685429	0,0
2	0,4772498680518208	0,4772498680518208	0,0
3	0,4986501019683699	0,4986501019683699	0,0
4	0,4999683287581669	0,4999683287581669	0,0
5	0,4999997133484281	0,49999971334842813	$1{,}1102236611198718 \times 10^{-16}$
6	0,4999999990134123	0,49999999901341236	$1{,}1102230268158213 \times 10^{-16}$
7	0,49999999999872013	0,4999999999987202	$1{,}1102230246279984 \times 10^{-16}$
8	0,49999999999999933	0,4999999999999994	$1{,}110223024625158 \times 10^{-16}$
9	0.5	0.5	0.0

Fuente: elaboración propia.

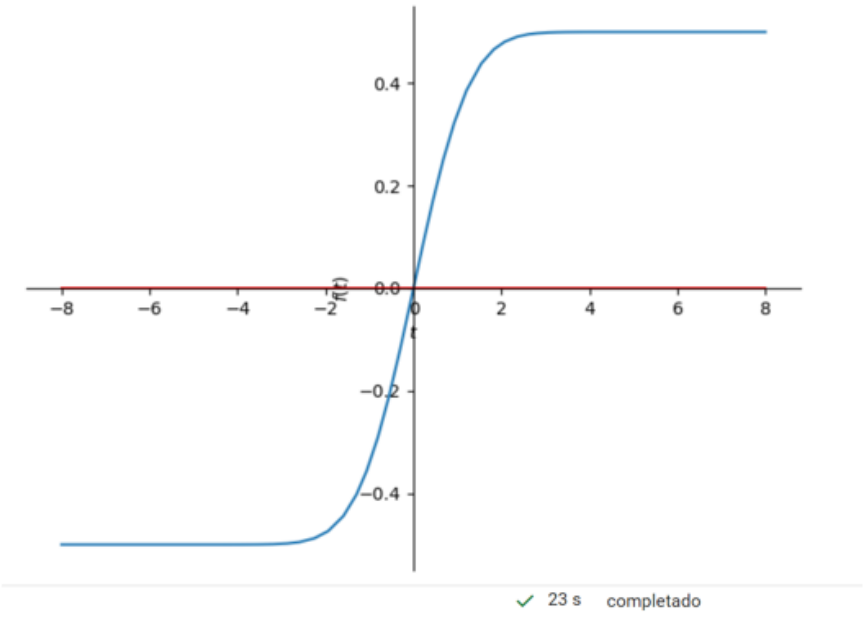
De los resultados obtenidos en la tabla 1, los datos de la columna del error relativo α_h son menores al indicador de prueba; es decir: todos los valores $\alpha_h \leq \alpha = 1.0\ X\ 10^{-15}$

Una representación gráfica de la función de distribución de probabilidad normal estándar se obtiene haciendo uso del instrumento, figura 5:

$$P[-8 \leq Z \leq 8] \approx \frac{1}{\sqrt{2\pi}} \int_{-8}^{8} \sum_{m=0}^{299} He_m(0) \frac{t^m}{m!} dt$$

Figura 9. Aproximación gráfica de la función de distribución de probabilidad normal estándar

```
1  Grafica(Hm(1,299)[4],0,t,-8,8)
```



✓ 23 s completado

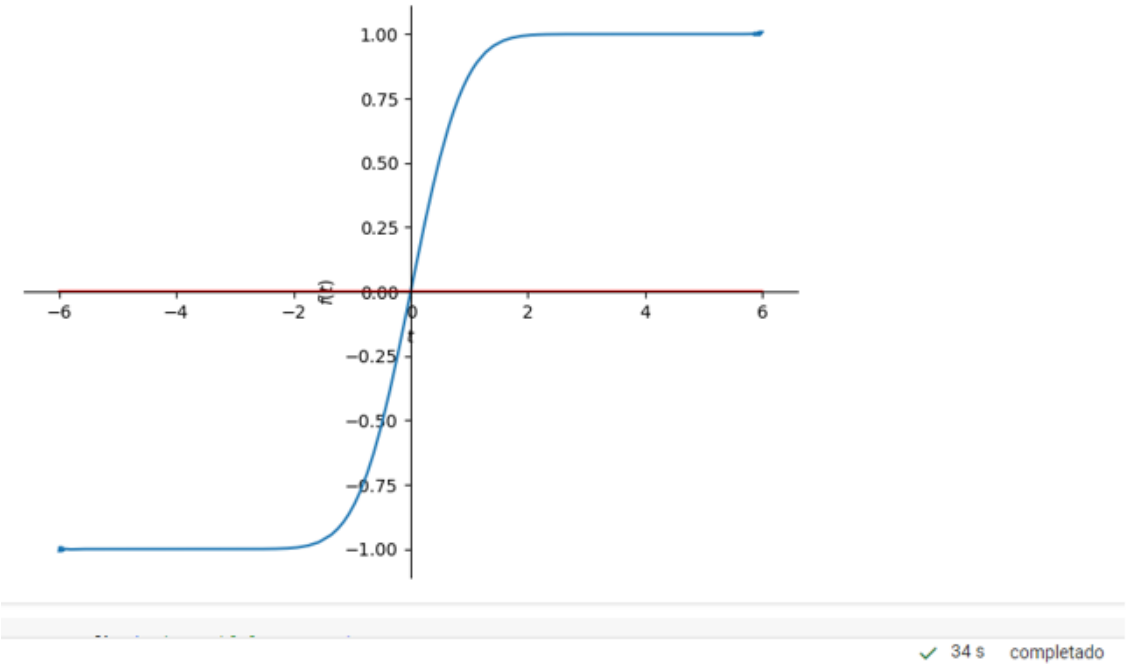
```
1  erf = Hm(2,302)[4]
2
3  z = -1
4  while z < 6:
5    z += 1
6    print("erf(",z,") =",float(erf.subs(t,z)))
7
8  Grafica(erf,0,t,-6,6)
```

```
erf( 0 ) = 0.0
erf( 1 ) = 0.8427007929497149
erf( 2 ) = 0.9953222650189527
erf( 3 ) = 0.9999779095030015
erf( 4 ) = 0.9999999845827421
erf( 5 ) = 0.9999999999984626
erf( 6 ) = 1.0
```

Fuente: elaboración propia.

De forma similar al anterior procedimiento, se realizaron aproximaciones numéricas y gráfica para la función $erf(-6 \leq z \leq 6)$:

Figura 10. Aproximaciones a la función $erf(t)$



Fuente: elaboración propia.

Comparando los valores dados en la figura 4, respecto a los dados en la figura 10, mediante la fórmula de error relativo, se tienen los siguientes valores α_h:

Tabla 2. Error relativo entre valores para $erf(z)$

h	X	Xh	Error relativo α_h
1	0.8427007929497149	0.8427007929497149	0,0
2	0.9953222650189527	0.9953222650189527	0,0
3	0.9999779095030014	0.9999779095030015	$1.1102475505453396 \times 10^{-16}$
4	0.9999999845827421	0.9999999845827421	0,0
5	0.9999999999984626	0.9999999999984626	0,0
6	1,0	1,0	0,0

Fuente: elaboración propia.

De los resultados obtenidos en la tabla 2, se encuentra que en las columnas del error relativo α_h, todos los valores son menores al indicador de precisión propuesto en la metodología, es decir que $\alpha_h \leq \alpha = 1.0\ X\ 10^{-15}$.

Análisis de la complejidad del algoritmos iterativo y recurrente

Duch, A. (2007) menciona que, para analizar la complejidad de un algoritmo iterativo, es útil desglosarlo en pasos y considerar la complejidad de cada uno:

Las operaciones de inicialización de variables como $m = 0, H = 1, fgxt = 1$ y $fac = 1$ son operaciones de tiempo constante, es decir, $O(1)$.

En cada iteración del bucle *while*, se realizan operaciones como el cálculo de derivadas, integrales y actualizaciones de variables. La complejidad de estas operaciones dependerá de las funciones utilizadas para el caso de polinomios de una variable. La derivación y la integración simbólica se realizan aplicando reglas algebraicas estándar, particularmente, cada término del polinomio implica cambiar el grado, numéricamente en uno, y multiplicar o dividir por el nuevo grado correspondientemente. Estas operaciones son computacionalmente eficientes, por lo que la complejidad es la del polinomio de mayor grado h, que coincide con la ejecución del bucle *"while"* hasta que m alcanza el valor de h, por lo que la complejidad es lineal $O(h)$.

Las operaciones después del bucle, y cálculos finales, son de tiempo constante. En general, la complejidad de este algoritmo es dominada por el bucle *"while"* y, por lo tanto, sigue siendo $O(h)$, donde h es el valor final de salida del bucle.

Sin embargo, se debe tener en cuenta que el instrumento de investigación da varios resultados, requiriendo mayor capacidad de almacenamiento o espacio, elemento que se puede mejorar aplicando el instrumento solo para lo que sea suficiente y necesario para la investigación. Se puede construir un algoritmo recurrente, que solo arroje la variable que se desea implementar. En el caso ejemplo de esta investigación y como se puede ver en el análisis numérico, solo se requiere guardar en una variable el valor de las funciones $He(0,t)$ o $H(0,t)$, según sea el tipo de polinomios de Hermite, es decir, de las definiciones 1 o 2. Un algoritmo recurrente puede ser el que sigue:

Figura 11. Algoritmo recurrente de integrales para obtener polinomios Hermite

```
def Hermite(H, fg0t, fac, m, tipo, p):          # Valores iniciales
  if m == p + 1:
    return tipo*fg0t/sqrt((3-tipo)*pi)         # Valor final
  else:
    dH = diff(H)                                # H'm(x)
    dH0 = - dH.subs(x,0)                        # H'm(0)
    H = tipo*m*integrate(H, x) + dH0            # Recurrencia Integral
    fac = fac*m                                 # Factorial
    fg0t = fg0t + dH0*t**m/fac                  # Función generadora de Hm(0)
    return Hermite(H, fg0t, fac, m+1, tipo, p) # Retorno recurrente

Hermite(1,1,1,1,1,302)                          # Llamada de la función
```

Fuente: elaboración propia.

Duch, A. (2007) también afirma que para analizar la complejidad del anterior algoritmo recursivo, es importante considerar la cantidad de operaciones realizadas en cada llamada recursiva y cuántas llamadas recursivas se realizan en total. La complejidad temporal se refiere a cómo crece el tiempo de ejecución en relación con el tamaño del problema.

En el caso base $m == p + 1$, la función simplemente devuelve el valor de tipo*fg0t/sqrt((3-tipo)*pi), cuyas operaciones son constantes, lo cual implica un tiempo constante $O(1)$.

Para el caso recursivo, se calcula la derivada dH del polinomio actual H; se evalúa dH en $x = 0$ para obtener dH0; se actualiza el polinomio de Hermite H mediante una integral y algunas operaciones aritméticas; se actualiza el factorial fac; se actualiza la función generadora fg0t; se realiza una llamada recursiva con parámetros actualizados. Cada actualización se da con operaciones aritméticas básicas de tiempo constante $O(1)$

Ahora, para analizar la recurrencia de la función, se tienen la ecuación:

$$T(m) = T(m + 1) + O(operaciones\ recursivas)$$

La función hace una llamada recursiva con m aumentado en 1 en cada iteración hasta que alcanza el caso p + 1. Por lo tanto, la cantidad total de llamadas recursivas es p + 1.

Para resolver esta recurrencia, se puede usar el método de sustitución. Suponga que:

$$T(m) = c \cdot m + g(m),$$

donde c es una constante determinada y g(m) es una función que representa el término de las operaciones recursivas. Sustituyendo esto en la ecuación de recurrencia:

$$c \cdot m + g(m) = c \cdot (m + 1) + O(operaciones\ recursivas)$$

Simplificando y resolviendo para $g\,(m)$:

$$g(m) = c + O(operaciones\ recursivas)$$

Esto implica que $g(m)$ es una función constante en términos de m, y la complejidad total de $T(m)$ es $O(m)$. En resumen, la solución aproximada de la recurrencia es:

$$T(m) = O(m).$$

Los resultados del análisis del algoritmo iterativo y del algoritmo recursivo coinciden con el tiempo de ejecución, evidenciando en la similitud de los tiempos de ejecución de los códigos expuestos en las figuras 7, 8, 10 y 11.

Estos tiempos de ejecución son finitos y relativamente bajos. Sin embargo, es posible mejorarlos guardando en una variable el resultado que arroje, por ejemplo, el polinomio de aproximación para los valores de la función de distribución de probabilidad normal estándar arrojado por el algoritmo recurrente, en su forma tipo string, esto es:

NormEst = (... + t**9/3456 - t**7/336 + t**5/40 - t**3/6 + t)/sqrt(2*pi)

Su ejecución es la siguiente:

Figura 12. Comando para obtener los valores de la distribución de probabilidad normal estándar

```
1  DistriNormEst = float(NormEstan.subs(t,9))
2  DistriNormEst
```

```
0.5
```

✓ 0 s completado

Fuente: elaboración propia.

Como se puede observar en la anterior imagen, el tiempo de ejecución tiende a 0 segundos.

Por los resultados obtenidos en las tablas 1 y 2, el análisis de complejidad de los algoritmos de iteración y recurrencia, y la posibilidad de construir instrumentos, funciones y comandos que arrojen valores simbólicos, numéricos y gráficos a través de las recurrencias integrales de polinomios de Hermite, se rechaza la hipótesis nula, propuesta en la metodología. Esto significa que mediante estas recurrencias integrales es posible estudiar computacional, científica y numéricamente el comportamiento de las funciones error, densidad, distribución de probabilidad y en general la integral de Gauss. Estos resultados se dan por la calidad de las aproximaciones de los valores, de las mencionadas funciones, relativos a los valores que arrojan los comandos de Python para el mismo propósito.

Discusión

Los polinomios de Hermite tienden a ser útiles para aproximar los objetos relacionados con ellos, como la función de distribución normal, la función error y la integral de Gauss, cuya precisión depende de la cantidad de términos que se sumen de la serie generadora de los mismos polinomios. Los ceros de los polinomios de mayor grado pueden capturar más detalles de la función que se quiere integrar, resultando ser una función suave que puede estimarse asintóticamente, lo que significa que a medida que el número de puntos aumenta, el error disminuye de manera predecible. Este comportamiento se puede modelar, pero el costo computacional también aumenta. Esto se puede solucionar alojando la suma de polinomios en una variable, evitando la generación de los mencionados objetos, cada vez que se requiere un valor numérico de objetos matemáticos.

Conclusiones

Los polinomios de Hermite, al estar íntimamente relacionados con la función $f(t)=e^{-t\wedge 2}$, son ideales para aproximar integrales que involucran funciones con este tipo de decaimiento exponencial, como la distribución normal estándar, la función error y la integral de Gauss. La recurrencia que genera estos polinomios permite una implementación numérica eficiente, en el que el uso de cuadraturas basadas en los ceros de los polinomios asegura una alta precisión con un número relativamente bajo de puntos de evaluación. A través de las recurrencias integrales que generan los polinomios de Hermite, se exhibe una convergencia exponencial cuando se utiliza para integrar funciones suaves como las que aparecen en el cálculo de la función error o la distribución normal. Este comportamiento de convergencia permite alcanzar

altos niveles de precisión con un número moderado de polinomios de Hermite, haciendo el método eficiente para aplicaciones numéricas en las que las integrales de funciones gaussianas son frecuentes. La fórmula de recurrencia permite generar rápidamente los términos necesarios para la cuadratura del área, reduciendo la complejidad computacional. Esta estructura de recurrencia facilita una codificación directa y numéricamente estable, lo que resulta en un cálculo eficiente y robusto de los polinomios hasta órdenes elevados.

Bibliografía

Academia Lab. (2023). Polinomios de Hermite. [Enciclopedia]. https://academia-lab.com/enciclopedia/polinomios-de-hermite/

Álvarez, R. (2001). Polinomios ortogonales: historia y aplicaciones. *Boletín de la Sociedad Española de Matemática Aplicada,* 18(1), 19-45. https://idus.us.es/bitstream/handle/11441/41709/Polinomios%20ortogonales%20historia%20y%20aplicaciones.pdf

Calvo, J. y Calvo, A. (2020). La mecánica cuántica y la necesidad de incluirla en los planes de estudio. REITUM. *Revista de la Escuela de Ingeniería y Tecnología de la Unimonserrate,* 1(1), 14-28. DOI: 10.29151/reit.n1a2

Carrillo, S., y Hurtado, M. (2021). Appell and Sheffer sequences: on their characterizations through functionals and examples. *Comptes Rendus Mathemátique, Tome 359, no 2, pp 205-217.* Doi: 10.5802/crmath.172.

Chapra, S. y Canale, R. (2007). Métodos numéricos para ingenieros. (5ª ed.). México D.F., México: MacGraw-Hill Interamericana. https://www.freelibros.net/matematicas/metodos-numericos-para-ingenieros-6ta-edicion-steven-c-chapra

Cruz, J. y Tetlalmatzi, M. (2015). Integral gaussiana y función error para todos. *Revista Miscelánea Matemática de la Sociedad Matemática Mexicana, 59*(2014), 77-89. https://miscelaneamatematica.org/download/tbl_articulos.pdf2.a71ee757a-8409b0a.353930362e706466.pdf

Duch, A. (2007). Análisis de Algoritmos. *Barcelona, Universidad Politécnica de Barcelona.* https://www.cs.upc.edu/~duch/home/duch/analisis.pdf

Gallo, M. (2023). La evolución de los computadores, transformando el mundo a través de la tecnología. REITUM. *Revista de la Escuela de Ingeniería y Tecnología de la Unimonserrate, 4*(1), 9-24. DOI: 10.29151/reit.n4a2.

Google Colaboratory (2020). Cuadernos de edición de código en entorno portátil. [Librerías]. https://isolution.pro/es/t/google-colab?alias=tutorial-de-google-colab

Hurtado Miguel. (2020). De las sumas de potencias a las sucesiones de Appell y su caracterización a través de funcionales. [Tesis de maestría]. *Universidad Sergio Arboleda.* http://hdl.handle.net/11232/1743

Olvera, M. A. C., Rodríguez, A. C., González, J. A. R., & Gutiérrez, A. C. V. (2014). Fundamentos de computación para ingenieros. *Grupo Editorial Patria.* https://books.google.com.co/books?id=Kt3hBAAAQBAJ&lpg=PP1&ots=HBm-a_6EG-&dq=Fundamentos%20de%20Computaci%C3%B3n%20Cient%C3%ADfica&lr&hl=es&pg=PR10#v=onepage&q=Fundamentos%20de%20Computaci%C3%B3n%20Cient%C3%ADfica&f=true

Ortega, C. (2023). Investigación aplicada: Definición, tipos y ejemplos. [Enciclopedia]. https://www.questionpro.com/blog/es/investigacion-aplicada/

Patarroyo, K. (2019). A digression on Hermite polynomials [Matemática] https://www.researchgate.net/publication/330212193_A_digression_on_Hermite_polynomials

Plaza Gálvez, Luis Fernando. (2016). Modelación matemática en ingeniería. IE Revista de investigación educativa de la REDIECH, 7(13), 47-57. https://www.researchgate.net/publication/346410146_Modelacion_matematica_en_ingenieria

Python numpy. (2022). Librerías y funciones basadas en los polinomios de Hermite. https://numpy.org/doc/stable/reference/generated/numpy.polynomial.hermite.Hermite.html

Python statistics (2022). Librerías para la estadística y probabilidad. https://docs.python.org/3/library/statistics.html

Python math.erf() Method (2022). Librería para obtener los valores de la función error de Gauss. https://www.w3schools.com/python/ref_math_erf.asp

Sanz, J. (2019). La cuadratura gaussiana según Gauss. *La Gaceta de la RSME, 1(22),* 101, 116. https://gaceta.rsme.es/abrir.php?id=1492

Steven M. Romano y Gian Carlo Rota (1978). The Umbral Calculus. Journals Advances in Mathematics, 2(27), 95-188. https://doi.org/10.1016/0001-8708(78)90087-7

Sympy. (2020). Lenguaje de programación simbólico para matemáticas. [Software]. https://isolution.pro/es/t/sympy?alias=tutorial-de-sympy

Wolfram Hermite (2023). Representación simbólica y numérica de los polinomios de Hermite y sus propiedades. [Software]. https://reference.wolfram.com/language/ref/HermiteH.html