

República Dominicana
UNIVERSIDAD TECNOLÓGICA DEL CIBAO ORIENTAL
Universidad Patrimonio de la Comunidad



Facultad de Postgrado y Educación Continuada
Departamento de Postgrado

Tesis de Maestría en Matemática

TÍTULO

Análisis comparativo de integrales definidas utilizando métodos numéricos.

SUSTENTANTE

Rosario Yissel Medrano Marte

ASESOR

José Miguel Gómez Guzmán, MA

Cotuí, Prov. Sánchez Ramírez, Rep. Dom.

Año Académico 2024.

TÍTULO

**ANÁLISIS COMPARATIVO DE INTEGRALES DEFINIDAS UTILIZANDO
MÉTODOS NUMÉRICOS.**

Declaración de Autoría

Los conceptos emitidos en el presente trabajo de investigación , es responsabilidad exclusiva de la sustentante del mismo. Las citas y consultas bibliográfica se han realizado respetando el derecho de autor, por lo que, se libera a la UTECO de cualquier responsabilidad, aunque se observa que para la reproducción total o parcial del mismo se requiere de su autorización.

Análisis comparativo de integrales definidas utilizando métodos numéricos.

Composición del tribunal evaluador

Dr. Rainier V. Sánchez C

Metodología

Dr. Manuel Aurelio Diloné Alvarados

Contenidos

MA. Nelson Sánchez

Redacción

Índice general

Declaración de Autoría	III
Agradecimientos	VII
Dedicatoria	IX
Resumen	X
CAPÍTULO I: INTRODUCCIÓN	1
1.1 Introducción	1
1.2 Antecedentes.	1
1.2.1 Antecedentes históricos.	1
1.2.2 Antecedentes académicos	2
1.3 Planteamiento del problema	3
1.4 Justificación	5
1.5 Objetivos	6
1.5.1 Objetivo general	6
1.5.2 Objetivos específicos	6
1.6 Alcances y límites	6
CAPÍTULO II: REVISIÓN DE LITERATURA	7
2.1 Introducción	7
2.2 Marco teórico	7
2.2.1 Definiciones	7
2.3 Propiedades de la integral definida	9
2.4 Teoremas	9
2.4.1 Teorema fundamental del cálculo	9
2.4.2 Primer teorema fundamental del cálculo	10
2.4.3 Segundo teorema fundamental del cálculo	10
2.5 Métodos numéricos para integrales definidas	11
2.5.1 Sumas de Riemann	11
2.6 Métodos basados en funciones de interpolación	11
2.6.1 Regla del Trapecio	11

2.6.2	Regla de Simpson 1/3	12
2.6.3	Regla de Simpson 3/8	13
2.7	Fórmulas de Newton-Cotes	14
2.7.1	Regla del rectángulo	14
2.7.2	Regla del punto medio	14
2.7.3	Reglas de Simpson	15
2.7.4	Propiedades generales	15
2.7.5	Análisis del error	15
2.8	Cuadratura de Gauss	16
CAPÍTULO III: DISEÑO METODOLÓGICO		18
3.1	Introducción	18
3.2	Marco metodológico	18
3.2.1	Diseño de la investigación	18
3.2.2	Tipo de investigación	18
3.2.3	Método de la investigación	19
3.2.4	Enfoque de la investigación	19
3.3	Procedimientos	20
CAPÍTULO IV: PRESENTACIÓN DE RESULTADOS		21
4.1	Introducción	21
4.2	Fundamentos teóricos de las fórmulas de Newton-Cotes y la cuadratura de Gauss	21
4.2.1	Fórmulas de Newton-Cotes	21
4.2.2	Cuadratura de Gauss	22
4.3	Implementación computacional de los métodos numéricos	23
4.3.1	Descripción del entorno de desarrollo	23
4.3.2	Implementación de las fórmulas de Newton-Cotes	24
4.3.3	Implementación de la cuadratura de Gauss	27
4.4	Diseño de experimentos y evaluación de desempeño	29
4.4.1	Selección de funciones de prueba	29

4.4.2	Precisión de los métodos	30
4.5	Discusión de resultados	36
4.5.1	Ventajas y desventajas de cada método	36
4.5.2	Recomendaciones de uso según el tipo de función	37
CAPÍTULO V: CONCLUSIONES		38
5.1	Introducción	38
5.2	Conclusiones	38
5.3	Líneas futuras de investigación	39
Bibliografía		40
Anexo		42
Anexo A: Entorno de programación de los métodos de integración		43

Agradecimientos

Quiero expresar mi más profundo agradecimiento a todas las personas que, de una manera u otra, han contribuido al logro de este importante paso en mi vida.

A Dios: Por darme la fortaleza, la sabiduría y la perseverancia necesarias para alcanzar esta meta. Sin su guía y bendiciones, nada de esto habría sido posible.

A mi familia: Por su amor incondicional, apoyo constante y comprensión durante todos estos años de estudio. Su aliento y confianza en mí han sido una fuente inagotable de motivación.

A mis maestros: Por su dedicación y conocimiento que han sembrado en mí la pasión por aprender y mejorar constantemente. Sus enseñanzas han sido fundamentales para mi formación profesional y personal.

A mi asesor: El Licenciado José Miguel Gómez, mi más sincero agradecimiento por su paciencia, orientación y valiosas recomendaciones a lo largo de este proceso. Su apoyo y compromiso han sido clave para la realización de esta tesis.

A Efraín Martínez: Gracias por haberme brindado las primeras herramientas necesarias para iniciar este camino.

A Manuel Arismendy Batista Villa: Gracias por su apoyo, su liderazgo y comprensión, los cuales han sido determinantes para llegar hasta aquí.

A mis compañeros de la maestría: Gracias por su amistad, camaradería y colaboración. Compartir esta experiencia con ustedes ha sido enriquecedor y gratificante. Sus perspectivas y apoyo han hecho de este viaje uno más llevadero y significativo.

A la Universidad Tecnológica del Cibao Oriental (UTECO): Por acogerme como estudiante de la licenciatura y ahora en la maestría. Su apoyo institucional ha sido fundamental

para mi desarrollo académico y profesional.

Al Ministerio de Educación Superior, Ciencia y Tecnología (MESCYT): Gracias por brindarme la oportunidad de estudiar mi maestría a través de su programa de becas. Su apoyo financiero y confianza en mi potencial han sido cruciales para la realización de este sueño.

A todos ustedes, mi gratitud eterna por haber sido parte esencial de esta etapa de mi vida. Este logro también es suyo.

Dedicatoria

A Dios: Por ser mi guía constante y fuente inagotable de fortaleza y sabiduría. Su presencia en mi vida ha sido fundamental para alcanzar este logro.

A mi madre: Divina María Marte, por su amor incondicional y su apoyo inquebrantable. Gracias por enseñarme el valor de la perseverancia y el esfuerzo.

A mi padre: Julio César Medrano, por su ejemplo de dedicación. Su apoyo y consejos han sido una inspiración constante para mí.

A mis hermanos: Por su compañía y aliento en cada etapa de mi vida. Su apoyo y amor han sido una fuente de motivación inestimable.

A mi esposo: Justo García Custodio, por su paciencia, comprensión y amor. Gracias por estar siempre a mi lado, apoyándome en cada paso de este camino.

A mis hijos: Jiana Yisbel García Medrano y Yismael García Medrano, por ser mi mayor inspiración y razón de ser. Su amor y alegría me han impulsado a alcanzar mis metas.

A todos ustedes dedico este trabajo con amor y gratitud infinita.

República Dominicana
UNIVERSIDAD TECNOLÓGICA DEL CIBAO ORIENTAL
UTECO

Facultad De Postgrado y Educación Continuada



Maestría en matemáticas

Título: Análisis comparativo de integrales definidas utilizando métodos numéricos.

Sustentante: Rosario Yissel Medrano Marte

Dirigido por: José Miguel Gómez Guzmán

RESUMEN

Este estudio presenta un análisis comparativo de métodos numéricos para el cálculo de integrales definidas, enfocándose en las fórmulas de Newton-Cotes y la cuadratura de Gauss. Se implementaron computacionalmente la regla del trapecio, las reglas de Simpson 1/3 y 3/8, y la cuadratura de Gauss-Legendre utilizando Python. Se evaluaron estos métodos con tres categorías de funciones: suaves, oscilantes y con singularidades, en el intervalo $[0, 2]$.

Los experimentos se realizaron variando el número de subintervalos o puntos de evaluación (mínimo requerido, aproximadamente 10 y 100) para cada método. Se midió la precisión mediante el error relativo y se registró el tiempo de ejecución.

Los resultados mostraron que para funciones suaves, todos los métodos convergieron, con la cuadratura de Gauss alcanzando la mayor precisión. En funciones oscilantes, las reglas de Simpson y la cuadratura de Gauss superaron significativamente a la regla del trapecio. Para funciones con singularidades, la cuadratura de Gauss generalmente proporcionó los mejores resultados, aunque todos los métodos requirieron un alto número de puntos para

lograr precisión aceptable.

La cuadratura de Gauss demostró la mayor precisión general, pero con tiempos de ejecución más largos. Las reglas de Simpson ofrecieron un buen equilibrio entre precisión y eficiencia computacional. La regla del trapecio, aunque menos precisa, mostró los tiempos de ejecución más cortos.

Se concluye que la selección del método óptimo depende de las características específicas de la función y los requisitos de precisión y eficiencia. Este estudio proporciona una guía para la selección e implementación de métodos de integración numérica en aplicaciones científicas y de ingeniería.

CAPÍTULO I INTRODUCCIÓN

1.1 Introducción

En este capítulo se presenta un conjunto de investigaciones que guardan estrecha relación con el tema de esta investigación y que funcionarán como marco de referencia para el desarrollo de este proyecto. Las investigaciones en cuestión constituyen los antecedentes históricos y académicos y hacen referencia a los métodos de la formula de Newton-Coste y la Cuadratura de Gauss, así como a su aplicación en la integrales definidas.

Además, este capítulo incluye el planteamiento del problema, los objetivos, la justificación de la investigación y los alcances del estudio, elementos que ilustran lo que se persigue conseguir con este proyecto.

1.2 Antecedentes.

1.2.1 Antecedentes históricos.

1.2.1.1 Origen y desarrollo de la integrales definidas, las formulas de Newton-Coste y la Cuadratura de Gauss

La integral definida tiene sus raíces en la antigua Grecia, donde matemáticos como Eudoxo y Arquímedes desarrollaron métodos para calcular áreas y volúmenes utilizando el método exhaustivo, que se basaba en aproximar estas cantidades mediante figuras geométricas conocidas (Fernández, 2016). Sin embargo, fue en el siglo XVII cuando se sentaron las bases del cálculo integral moderno, con los trabajos de Isaac Newton y Gottfried Leibniz. Newton introdujo el concepto de fluentes y fluxiones, mientras que Leibniz desarrolló la notación y el enfoque basado en sumas infinitesimales, sentando las bases para la formulación rigurosa del concepto de integral definida (Pérez, 2018).

En el siglo XIX, Augustin-Louis Cauchy y Bernhard Riemann formularon defini-

ciones precisas de la integral definida utilizando límites y sumas, conocidas como sumas de Riemann (Gutiérrez, 2017). Estas definiciones sentaron las bases para el desarrollo posterior de la teoría de integración y sus aplicaciones.

Paralelamente al desarrollo del cálculo integral, surgió la necesidad de encontrar métodos efectivos para aproximar el valor de las integrales definidas cuando su cálculo exacto no era posible. En este contexto, se desarrollaron las fórmulas de Newton-Cotes, nombradas así en honor a Isaac Newton y Roger Cotes. Estas fórmulas, que incluyen las reglas del rectángulo, del trapecio y de Simpson, aproximan la integral definida utilizando polinomios de interpolación (Sánchez, 2015).

Otro método de aproximación que ganó relevancia fue la cuadratura de Gauss, desarrollada por Carl Friedrich Gauss a principios del siglo XIX. Este método se basa en aproximar la integral definida utilizando una suma ponderada de valores de la función en puntos específicos, conocidos como nodos de cuadratura (Fernández, 2016). La cuadratura de Gauss demostró ser altamente eficiente y precisa en comparación con otros métodos de la época.

1.2.2 Antecedentes académicos

En el ámbito académico, la integral definida y los métodos numéricos para su cálculo han sido objeto de estudio y análisis exhaustivo. Diversos autores han contribuido al desarrollo y mejora de estos métodos, así como a su aplicación en diferentes campos de la ciencia y la ingeniería.

En su trabajo seminal, Davis y Rabinowitz (1984) presentaron un análisis detallado de los métodos de integración numérica, incluyendo las fórmulas de Newton-Cotes y la cuadratura de Gauss. Su obra sentó las bases para la comprensión y aplicación de estos métodos en la resolución de problemas prácticos.

Posteriormente, autores como Burden y Faires (2011) y Chapra y Canale (2015) han presentado tratamientos exhaustivos de los métodos numéricos para la integración, incluyendo análisis de error, convergencia y estabilidad. Sus obras han sido ampliamente utilizadas como referencias en cursos universitarios de análisis numérico.

En años recientes, se han realizado investigaciones para mejorar la eficiencia y precisión de los métodos de integración numérica. Por ejemplo, Gander y Gautschi (2000) propusieron una modificación adaptativa de la cuadratura de Gauss para manejar integrales con singularidades y oscilaciones. Asimismo, se han desarrollado métodos basados en transformadas wavelets (Sweldens y Piessens, 1994) y técnicas de integración multidimensional (Cools, 1997) para abordar problemas más complejos.

1.3 Planteamiento del problema

La integración numérica desempeña un papel fundamental en diversas áreas de la ciencia y la ingeniería, donde a menudo se requiere calcular integrales definidas que no tienen una solución analítica cerrada. En estos casos, es necesario recurrir a métodos numéricos que aproximen el valor de la integral con una precisión aceptable.

Sin embargo, la elección del método numérico adecuado no siempre es trivial y depende de las características de la función a integrar, el intervalo de integración y los requisitos de precisión y eficiencia computacional. Cada método tiene sus propias ventajas y limitaciones, y comprender su comportamiento y rendimiento es esencial para su aplicación efectiva.

En este contexto, surge la necesidad de realizar un análisis comparativo exhaustivo de los diferentes métodos numéricos para el cálculo de integrales definidas, con énfasis en las fórmulas de Newton-Cotes y la cuadratura de Gauss. Este análisis debe considerar aspectos como la precisión, la velocidad de convergencia, la estabilidad numérica y la capacidad para manejar funciones con características especiales, como discontinuidades o

singularidades.

Además, es importante investigar la aplicabilidad de estos métodos en problemas prácticos, considerando factores como la complejidad de implementación, el costo computacional y la adaptabilidad a diferentes escenarios. La identificación de las fortalezas y debilidades de cada método permitirá a los profesionales y investigadores seleccionar el enfoque más adecuado para sus necesidades específicas. Es por eso que se le buscar dar respuesta a las siguientes interrogantes:

- ¿Cuáles son los fundamentos teóricos de las fórmulas de Newton-Cotes y la cuadratura de Gauss, incluyendo sus derivaciones, propiedades y análisis de error?
- . ¿Cómo se pueden implementar los métodos numéricos seleccionados utilizando un lenguaje de programación adecuado y qué resultados se obtienen al realizar experimentos computacionales para evaluar su desempeño en diferentes escenarios?
- ¿Cómo se comparan la precisión y velocidad de convergencia de los métodos numéricos al aplicarlos a una variedad de funciones con características diversas, como suavidad, oscilaciones y singularidades?

Para abordar estas preguntas, se realizaron experimentos numéricos con diferentes integrales definidas, variando condiciones iniciales y parámetros relevantes. Los resultados se analizarán cuantitativa y cualitativamente, comparando convergencia, precisión, estabilidad y coste computacional. Además, se discutirán implicaciones prácticas, incluyendo aplicaciones potenciales en ingeniería, física, química y otras disciplinas donde sea necesario resolver integrales definidas complejas de manera precisa y eficiente.

Este análisis busca brindar una comprensión profunda de las fortalezas y limitaciones de los métodos de de la formula de Newton-Coste y la cuadratura de Gauss, guiando la selección del método más adecuado según el contexto y la aplicación para resolver integrales definidas.

1.4 Justificación

El análisis comparativo de los métodos numéricos para el cálculo de integrales definidas tiene una gran relevancia tanto en el ámbito académico como en la práctica profesional. Desde una perspectiva teórica, este estudio contribuye al conocimiento fundamental de los métodos numéricos y sus propiedades, brindando una comprensión más profunda de su comportamiento y limitaciones.

Además, los resultados de este análisis tienen implicaciones prácticas significativas. Los profesionales e investigadores de diversas disciplinas, como la física, la ingeniería y las ciencias aplicadas, a menudo se enfrentan a problemas que requieren la evaluación de integrales definidas. Contar con un conocimiento sólido de los métodos numéricos disponibles y sus características les permitirá tomar decisiones informadas sobre qué enfoque utilizar en función de las particularidades de cada problema.

La selección adecuada del método numérico puede tener un impacto directo en la precisión de los resultados, la eficiencia computacional y la confiabilidad de las simulaciones y modelos matemáticos. Por lo tanto, este estudio proporcionará a los profesionales herramientas y directrices valiosas para optimizar sus cálculos y obtener resultados más precisos y confiables.

Además, el análisis comparativo propuesto puede impulsar futuras investigaciones en el campo de los métodos numéricos, identificando áreas de mejora y oportunidades para el desarrollo de nuevas técnicas y algoritmos. Esto, a su vez, contribuirá al avance general del conocimiento y las capacidades en el ámbito del análisis numérico.

El presente estudio se justifica por su contribución tanto al conocimiento teórico como a la práctica profesional, brindando una comprensión más profunda de los métodos numéricos para el cálculo de integrales definidas y proporcionando herramientas y directrices valiosas para su aplicación efectiva en problemas reales.

1.5 Objetivos

1.5.1 Objetivo general

Realizar un análisis comparativo exhaustivo y riguroso de los métodos numéricos empleados para el cálculo de integrales definidas.

1.5.2 Objetivos específicos

- Describir los fundamentos teóricos de las fórmulas de Newton-Cotes y la cuadratura de Gauss, incluyendo sus derivaciones, propiedades y análisis de error.
- Implementar los métodos numéricos seleccionados utilizando un lenguaje de programación adecuado y realizar experimentos computacionales para evaluar su desempeño en diferentes escenarios.
- Comparar la precisión y velocidad de convergencia de los métodos numéricos al aplicarlos a una variedad de funciones con características diversas, como suavidad, oscilaciones y singularidades.

1.6 Alcances y límites

El presente estudio se centrará en el análisis comparativo de los métodos numéricos para el cálculo de integrales definidas, con énfasis en las fórmulas de Newton-Cotes y la cuadratura de Gauss. Se considerarán funciones unidimensionales y se evaluará el desempeño de los métodos en términos de precisión, velocidad de convergencia, estabilidad numérica y aplicabilidad en problemas prácticos.

Se utilizarán implementaciones en un lenguaje de programación adecuado y se realizarán experimentos computacionales para respaldar los análisis y comparaciones. Sin embargo, no se desarrollarán nuevos métodos numéricos ni se abordarán extensiones a integrales multidimensionales o estocásticas.

CAPÍTULO II

REVISIÓN DE LITERATURA

2.1 Introducción

En este capítulo se presentan los conceptos fundamentales y las definiciones clave relacionados con la integración numérica y los métodos para aproximar integrales definidas. Se abordan temas como la integral definida, los métodos numéricos, la precisión, la aproximación, el error de aproximación, las sumas de Riemann, el análisis numérico, las fórmulas de Newton-Cotes y la cuadratura de Gauss. Estos conceptos sientan las bases teóricas necesarias para comprender y analizar los métodos numéricos que se discutirán en los capítulos posteriores.

2.2 Marco teórico

2.2.1 Definiciones

Integral definida: La integral definida de una función $f(X)$ en u intervalo $[a, b]$ se define como el limite de las sumas de Riemann cuando el numero de subintervalos tiende a infinitos y la norma de la particion se aproxima a cero (Larson y Edwards, 2016). Se denota como $\int_a^b f(x) dx$ y presenta el area bajo la curva de $f(x)$ en el intervalo $[a, b]$.

Métodos numéricos: Los métodos numéricos son técnicas y algoritmos utilizados para aproximar soluciones de problemas matemáticos cuando no es posible obtener una solución analítica exacta (Burden y Faires, 2011). En el contexto de la integración, los métodos numéricos se emplean para aproximar el valor de una integral definida.

Precisión: La precisión se refiere a la proximidad entre el valor aproximado obtenido mediante un método numérico y el valor exacto de la integral definida (Chapra y Canale, 2015). Cuanto menor sea la diferencia entre estos valores, mayor será la precisión del método.

Aproximación: Una aproximación es un valor estimado que se obtiene mediante un método numérico y que se utiliza en lugar del valor exacto de una cantidad matemática, como una integral definida (Burden y Faires, 2011). Las aproximaciones son necesarias cuando no es posible obtener una solución exacta o cuando el cálculo de la solución exacta es computacionalmente costoso.

Error de aproximación: El error de aproximación es la diferencia entre el valor aproximado obtenido mediante un método numérico y el valor exacto de la cantidad que se está aproximando (Chapra y Canale, 2015). En el caso de la integración numérica, el error de aproximación mide la discrepancia entre la aproximación numérica de la integral y su valor real.

Sumas de Riemann: Las sumas de Riemann son una construcción matemática utilizada para aproximar el valor de una integral definida (Larson y Edwards, 2016). Consisten en dividir el intervalo de integración en subintervalos y aproximar el área bajo la curva en cada subintervalo mediante rectángulos. La suma de las áreas de estos rectángulos proporciona una aproximación de la integral.

Análisis numérico: El análisis numérico es una rama de las matemáticas que se ocupa del desarrollo y análisis de algoritmos para la resolución aproximada de problemas matemáticos (Burden y Faires, 2011). Incluye el estudio de la precisión, la estabilidad y la eficiencia de los métodos numéricos.

Fórmulas de Newton-Cotes: Las fórmulas de Newton-Cotes son un conjunto de métodos de integración numérica que aproximan la integral definida utilizando polinomios de interpolación (Davis y Rabinowitz, 1984). Estas fórmulas incluyen las reglas del rectángulo, del trapecio y de Simpson, entre otras.

Cuadratura de Gauss: La cuadratura de Gauss es un método de integración numérica que aproxima la integral definida utilizando una suma ponderada de valores de la función en puntos específicos, conocidos como nodos de cuadratura (Davis y Rabinowitz, 1984). Este método se caracteriza por su alta precisión y eficiencia en comparación con otros métodos de integración numérica.

2.3 Propiedades de la integral definida

La integral definida posee varias propiedades fundamentales que son relevantes para su cálculo y análisis (Larson y Edwards, 2016):

1. Linealidad: Si f y g son funciones integrables en $[a, b]$ y c es una constante, entonces:

$$\int_a^b (f(x) + g(x)) dx = \int_a^b f(x) dx + \int_a^b g(x) dx$$

$$\int_a^b c \cdot f(x) dx = c \cdot \int_a^b f(x) dx$$

2. Aditividad: Si f es integrable en $[a, b]$ y c es un punto en el intervalo $[a, b]$, entonces:

$$\int_a^b f(x) dx = \int_a^c f(x) dx + \int_c^b f(x) dx$$

3. Monotonía: Si f y g son funciones integrables en $[a, b]$ y $f(x) \leq g(x)$ para todo x en $[a, b]$, entonces:

$$\int_a^b f(x) dx \leq \int_a^b g(x) dx$$

4. Valor absoluto: Si f es integrable en $[a, b]$, entonces:

$$\left| \int_a^b f(x) dx \right| \leq \int_a^b |f(x)| dx$$

5. Teorema del valor medio: Si f es continua en $[a, b]$, entonces existe un punto c en (a, b) tal que:

$$\int_a^b f(x) dx = f(c) \cdot (b - a)$$

Estas propiedades son útiles para manipular y simplificar expresiones que involucren integrales definidas, así como para establecer relaciones y desigualdades entre ellas.

2.4 Teoremas

2.4.1 Teorema fundamental del cálculo

El teorema fundamental del cálculo establece una conexión crucial entre las operaciones de derivación e integración (Larson y Edwards, 2016). Consta de dos partes principales:

1. Primera parte: Si f es una función continua en $[a, b]$ y F es una antiderivada de f en $[a, b]$, entonces:

$$\int_a^b f(x) dx = F(b) - F(a)$$

2. Segunda parte: Si f es una función continua en $[a, b]$ y se define una función F como:

$$F(x) = \int_a^b f(x) dt$$

entonces F es una antiderivada de f en $[a, b]$, es decir, $F'(x) = f(x)$ para todo x en $[a, b]$.

El teorema fundamental del cálculo permite evaluar integrales definidas utilizando antiderivadas y establece una relación inversa entre la derivación y la integración.

2.4.2 Primer teorema fundamental del cálculo

El primer teorema fundamental del cálculo, también conocido como el teorema de Newton-Leibniz, establece que si f es una función continua en $[a, b]$ y F es una antiderivada de f en $[a, b]$, entonces:

$$\int_a^b f(x) dx = F(b) - F(a)$$

Este teorema proporciona una forma de calcular la integral definida de una función continua utilizando cualquier antiderivada de la función, evaluada en los límites de integración.

2.4.3 Segundo teorema fundamental del cálculo

El segundo teorema fundamental del cálculo establece que si f es una función continua en $[a, b]$ y se define una función F como:

$$F(x) = \int_x^b f(t) dt$$

entonces F es una antiderivada de f en $[a, b]$, es decir, $F'(x) = f(x)$ para todo x en $[a, b]$.

Este teorema muestra que la integración y la derivación son operaciones inversas entre sí, y que la integral definida puede utilizarse para construir antiderivadas de funciones continuas.

2.5 Métodos numéricos para integrales definidas

2.5.1 Sumas de Riemann

Definición: Las sumas de Riemann son una construcción matemática utilizada para aproximar el valor de una integral definida. Consisten en dividir el intervalo de integración $[a, b]$ en n subintervalos de igual longitud, y aproximar el área bajo la curva de la función $f(x)$ en cada subintervalo mediante rectángulos (Larson y Edwards, 2016).

Propiedades: Las sumas de Riemann tienen las siguientes propiedades:

- A medida que el número de subintervalos n aumenta, la aproximación de la integral mejora.
- El límite de las sumas de Riemann cuando n tiende a infinito es igual al valor exacto de la integral definida.
- La elección de los puntos de muestra en cada subintervalo (punto inicial, punto medio o punto final) afecta la precisión de la aproximación.

Ejemplos de aplicación: Las sumas de Riemann se utilizan como base para desarrollar métodos de integración numérica más avanzados, como las reglas del trapecio y de Simpson. También se emplean para ilustrar conceptualmente el proceso de aproximación de integrales definidas.

2.6 Métodos basados en funciones de interpolación

2.6.1 Regla del Trapecio

Definición: La regla del trapecio es un método de integración numérica que aproxima la integral definida de una función utilizando trapecios en lugar de rectángulos. Consiste en dividir el intervalo de integración $[a, b]$ en n subintervalos de igual longitud y aproximar el área bajo la curva en cada subintervalo mediante un trapecio (Burden y Faires, 2011).

Fórmula: La fórmula de la regla del trapecio para aproximar la integral definida $\int_a^b f(x) dx$ es: $\int_a^b f(x) dx \approx (h/2) \cdot [f(a) + 2 \cdot \sum_{i=1}^{n-1} f(x_i) + f(b)]$

donde $h = (b - a)/n$ es la longitud de cada subintervalo y $x_i = a + i.h$ son los puntos de division.

Análisis del error: El error de aproximación de la regla del trapecio es proporcional a h^2 , es decir, disminuye cuadráticamente a medida que el número de subintervalos n aumenta. La fórmula del error es:

$$|E| \leq (b - a)^3 \cdot M / (12n^2)$$

donde M es una cota superior por la segunda derivada de f en $[a, b]$.

Ejemplos de aplicación: La regla del trapecio se utiliza comúnmente para aproximar integrales definidas en una amplia variedad de aplicaciones, como el cálculo de áreas, volúmenes y longitudes de curvas. Es especialmente útil cuando la función es relativamente suave y no presenta oscilaciones rápidas.

2.6.2 Regla de Simpson 1/3

Definición: La regla de Simpson 1/3 es un método de integración numérica que aproxima la integral definida de una función utilizando polinomios cuadráticos. Consiste en dividir el intervalo de integración $[a, b]$ en un número par de subintervalos de igual longitud y aproximar el área bajo la curva en cada par de subintervalos mediante un polinomio cuadrático (Chapra y Canale, 2015).

Fórmula: La fórmula de la regla de Simpson 1/3 para aproximar la integral definida $\int_a^b f(x) dx$ es:

$$\int_a^b f(x) dx \approx (h/3) \cdot [f(a) + 4 \cdot \sum_{i=1}^{n-1} f(x_{2i-1}) + 2 \cdot \sum_{i=1}^{n-1} f(x_{2i}) + f(b)]$$

donde $h = \frac{(b-a)}{n}$ es la longitud de cada subintervalo, n es un numero par y $x_i = a + i.h$ son los puntos de división.

Análisis del error: El error de aproximación de la regla de Simpson 1/3 es proporcional a h^4 , es decir, disminuye rápidamente a medida que el número de subintervalos n aumenta. La fórmula del error es:

$$|E| \leq (b - a)^5 \cdot M / (180n^4)$$

donde M es una cota superior para la cuarta derivada de f en $[a, b]$.

Ejemplos de aplicación: La regla de Simpson 1/3 se utiliza ampliamente para aproximar integrales definidas en diversos campos, como la física, la ingeniería y las ciencias aplicadas. Es especialmente efectiva cuando la función es suave y bien comportada en el intervalo de integración.

2.6.3 Regla de Simpson 3/8

Definición: La regla de Simpson 3/8 es una variante de la regla de Simpson que utiliza polinomios cúbicos para aproximar la integral definida. Consiste en dividir el intervalo de integración $[a, b]$ en un número múltiplo de 3 subintervalos de igual longitud y aproximar el área bajo la curva en cada grupo de tres subintervalos mediante un polinomio cúbico (Burden y Faires, 2011).

Fórmula: La fórmula de la regla de Simpson 3/8 para aproximar la integral definida

$$\int_a^b f(x) dx \text{ es:}$$

$$\int_a^b f(x) dx \approx (3h/8) \cdot [f(a) + 3 \cdot \sum_{i=1}^{n-1} f(x_{3i-2}) + 3 \cdot \sum_{i=1}^{n-1} f(x_{3i-1}) + 2 \cdot \sum_{i=1}^{n-1} f(x_{3i}) + f(b)]$$

donde $h = \frac{(b-a)}{n}$ es la longitud de cada subintervalo, n es un múltiplo de 3, y $x_i = a + i \cdot h$ son los puntos de división.

Análisis del error: El error de aproximación de la regla de Simpson 3/8 es proporcional a h^4 , al igual que la regla de Simpson 1/3. La fórmula del error es:

$$|E| \leq (b - a)^5 \cdot M / (80n^4)$$

donde M es una cota superior para la cuarta derivada de f en $[a, b]$.

Ejemplos de aplicación: La regla de Simpson 3/8 se utiliza en situaciones similares a la regla de Simpson 1/3, especialmente cuando se requiere una mayor precisión. Es particularmente útil cuando la función tiene un comportamiento suave y se pueden utilizar polinomios cúbicos para aproximarla de manera efectiva.

2.7 Fórmulas de Newton-Cotes

2.7.1 Regla del rectángulo

Definición: La regla del rectángulo es un método de integración numérica que aproxima la integral definida de una función utilizando rectángulos. Consiste en dividir el intervalo de integración $[a, b]$ en n subintervalos de igual longitud y aproximar el área bajo la curva en cada subintervalo mediante un rectángulo cuya altura es el valor de la función en el punto inicial o final del subintervalo (Burden y Faires, 2011).

Fórmulas:

- Regla del rectángulo a la izquierda: $\int_a^b f(x) dx \approx h \cdot \sum_{i=0}^{n-1} f(x_i)$
- Regla del rectángulo a la derecha: $\int_a^b f(x) dx \approx h \cdot \sum_{i=1}^n f(x_i)$

donde $h = (b - a)/n$ es la longitud de cada subintervalo y $x_i = a + i \cdot h$ son los puntos de división.

2.7.2 Regla del punto medio

Definición: La regla del punto medio es un método de integración numérica que aproxima la integral definida de una función utilizando rectángulos cuya altura es el valor de la función en el punto medio de cada subintervalo. Consiste en dividir el intervalo de integración $[a, b]$ en n subintervalos de igual longitud y aproximar el área bajo la curva en cada subintervalo mediante un rectángulo centrado en el punto medio (Chapra y Canale, 2015).

Fórmula: La fórmula de la regla del punto medio para aproximar la integral definida $\int_a^b f(x) dx$ es

$$\int_a^b f(x) dx \approx h \cdot \sum_{i=1}^n f(x_{i-1/2})$$

donde $h = (b - a)/n$ es la longitud de cada subintervalo y $(x_{i-1/2}) = (x_{i-1} + x_i)/2$ son los puntos medios de cada subintervalo.

2.7.3 Reglas de Simpson

Las reglas de Simpson, como la regla de Simpson 1/3 y la regla de Simpson 3/8, son casos especiales de las fórmulas de Newton-Cotes. Estas reglas utilizan polinomios de grado superior para aproximar la integral definida y lograr una mayor precisión en comparación con las reglas del rectángulo y del punto medio.

2.7.4 Propiedades generales

Las fórmulas de Newton-Cotes tienen las siguientes propiedades generales (Davis y Rabinowitz, 1984):

Son métodos de integración numérica basados en la interpolación polinómica.

Aproximan la integral definida utilizando una combinación lineal de los valores de la función en puntos específicos del intervalo de integración.

A medida que el grado del polinomio de interpolación aumenta, la precisión de la aproximación mejora, pero también aumenta el costo computacional.

Las fórmulas de Newton-Cotes cerradas, como las reglas del trapecio y de Simpson, utilizan los valores de la función en los extremos del intervalo de integración, mientras que las fórmulas abiertas no los utilizan.

2.7.5 Análisis del error

El análisis del error de las fórmulas de Newton-Cotes se basa en la teoría de la interpolación polinómica y el teorema del valor medio para integrales. El error de aproximación depende del grado del polinomio de interpolación utilizado y de las derivadas de la función en el intervalo de integración (Burden y Faires, 2011).

En general, el error de aproximación de las fórmulas de Newton-Cotes es proporcional a una potencia del tamaño de los subintervalos h . Cuanto mayor sea el grado del polinomio de interpolación, mayor será la potencia de h en la fórmula del error y, por lo tanto, más rápida será la convergencia del método.

2.8 Cuadratura de Gauss

Definición: La cuadratura de Gauss es un método de integración numérica que aproxima la integral definida de una función utilizando una suma ponderada de los valores de la función en puntos específicos, llamados nodos de cuadratura. A diferencia de las fórmulas de Newton-Cotes, que utilizan puntos igualmente espaciados, la cuadratura de Gauss selecciona los nodos de manera óptima para maximizar la precisión (Burden y Faires, 2011).

Fórmulas: La fórmula general de la cuadratura de Gauss para aproximar la integral definida $\int_a^b f(x) dx$ es

$$\int_a^b f(x) dx \approx \sum_{i=1}^n W_i \cdot f(x_i)$$

donde n es el número de nodos de cuadratura, x_i son los nodos de cuadratura y w_i son los pesos correspondientes. Los nodos y pesos se determinan de manera óptima para maximizar la precisión de la aproximación.

Análisis del error: El error de aproximación de la cuadratura de Gauss depende del número de nodos utilizados y de las propiedades de la función. En general, la cuadratura de Gauss puede alcanzar una precisión muy alta con un número relativamente pequeño de nodos en comparación con las fórmulas de Newton-Cotes (Davis y Rabinowitz, 1984).

Comparación con Newton-Cotes: La cuadratura de Gauss tiene varias ventajas sobre las fórmulas de Newton-Cotes:

- Mayor precisión: Para un número dado de nodos, la cuadratura de Gauss generalmente proporciona una aproximación más precisa que las fórmulas de Newton-Cotes.

- Flexibilidad: La cuadratura de Gauss puede adaptarse a diferentes intervalos de integración y funciones de peso, lo que permite su aplicación en una amplia variedad de problemas.
- Estabilidad numérica: La cuadratura de Gauss es menos propensa a errores de redondeo y problemas de estabilidad numérica en comparación con algunas fórmulas de Newton-Cotes de alto grado

CAPÍTULO III METODOLÓGIA

3.1 Introducción

En este capítulo se presenta el diseño metodológico del proyecto, que incluye el tipo y enfoque de la investigación, el método utilizado, las etapas de la investigación, las técnicas e instrumentos de recolección de datos, el análisis de datos y los procedimientos a seguir. Se detallarán los pasos para la implementación de los métodos de Halley y Chebyshev, el diseño de experimentos numéricos, la evaluación de convergencia, precisión y estabilidad, la comparación de los métodos y la validación y análisis de resultados.

3.2 Marco metodológico

3.2.1 Diseño de la investigación

Según Hernández Sampieri et al. (2014), esta investigación se clasifica como no experimental, transversal y analítica. Es no experimental porque no se manipulan las variables de estudio, sino que se realiza un análisis comparativo basado en la teoría existente y la implementación computacional de los métodos numéricos seleccionados para el cálculo de integrales definidas. Es transversal porque el análisis se realiza en un momento específico, sin implicar un seguimiento a lo largo del tiempo. Se considera analítica porque se examina minuciosamente el rendimiento de las fórmulas de Newton-Cotes y la cuadratura de Gauss, descomponiendo su funcionamiento en aspectos como precisión, velocidad de convergencia y estabilidad numérica, para así observar y comprender la naturaleza de estos métodos en la aproximación de integrales definidas con diferentes características, como funciones suaves, oscilantes y con singularidades.

3.2.2 Tipo de investigación

La investigación es de tipo descriptiva y analítica (Hernández Sampieri, 2014). Es descriptiva porque se detallan las características y propiedades de los métodos numéricos estudiados, así como sus fundamentos teóricos y matemáticos. Es analítica porque se rea-

liza un análisis comparativo de los métodos numéricos, evaluando su precisión, velocidad de convergencia, estabilidad numérica y aplicabilidad en diferentes escenarios.

3.2.3 Método de la investigación

El método utilizado es principalmente cualitativo, debido a que se efectuó la recopilación y análisis de información teórica y conceptual relacionada con los métodos numéricos para el cálculo de integrales definidas. Sin embargo, también incorpora elementos cuantitativos en la implementación computacional y evaluación del desempeño de los métodos mediante experimentos numéricos (Hernández Sampieri, 2014).

3.2.4 Enfoque de la investigación

El enfoque de esta investigación es cualitativo, ya que se recolectan y analizan datos numéricos para evaluar y comparar el rendimiento de los métodos numéricos, específicamente las fórmulas de Newton-Cotes y la Cuadratura de Gauss (Hernández Sampieri, 2014).

3.2.4.1 Etapas de la investigación

1. Revisión bibliográfica: Se realizó una revisión exhaustiva de la literatura relacionada con los métodos de las fórmulas de Newton-Cotes y la Cuadratura de Gauss, así como otros métodos iterativos para resolver integrales definidas.
2. Implementación de los métodos: Se implementarán los métodos de las fórmulas de Newton-Cotes y la Cuadratura de Gauss en un lenguaje de programación adecuado, como Python, siguiendo sus fórmulas iterativas y consideraciones numéricas.
3. Diseño de experimentos numéricos: Se diseñará un conjunto de experimentos que cubran diferentes tipos de integrales definidas, condiciones iniciales y escenarios de prueba para evaluar el rendimiento de los métodos.
4. Ejecución de experimentos y recolección de datos: Se ejecutarán los experimentos diseñados y se recolectarán datos sobre la convergencia, precisión, estabilidad numérica y eficiencia computacional de los métodos.

5. Análisis de resultados: Se analizarán los datos recolectados utilizando técnicas estadísticas para comparar el rendimiento de los métodos de las formulas de Newton-Coste y cuadratura de Gauss en diferentes escenarios.
6. Conclusiones y recomendaciones: Se presentarán las conclusiones del análisis comparativo y se brindarán recomendaciones sobre la selección del método más adecuado según el tipo de integrales y los requisitos de precisión y eficiencia.

3.3 Procedimientos

Implementación de los métodos de las formulas de Newton-Coste y la Cuadratura de Gauss

- Seleccionar un lenguaje de programación adecuado, como Python.
- Implementar las fórmulas iterativas de los métodos las formulas de Newton-Coste y la Cuadratura de Gauss, considerando el cálculo de las integrales y el criterio de parada.
- Verificar la correcta implementación de los métodos mediante pruebas unitarias con integrales definidas de referencia.

CAPÍTULO IV PRESENTACIÓN DE RESULTADOS

4.1 Introducción

En este capítulo se presentan los resultados obtenidos del análisis comparativo de los métodos numéricos para el cálculo de integrales definidas, enfocándose en las fórmulas de Newton-Cotes y la cuadratura de Gauss. Se abordan los fundamentos teóricos, la implementación computacional y la evaluación del desempeño de estos métodos en diversos escenarios.

4.2 Fundamentos teóricos de las fórmulas de Newton-Cotes y la cuadratura de Gauss

4.2.1 Fórmulas de Newton-Cotes

Derivación matemática:

Las fórmulas de Newton-Cotes se derivan a partir de la aproximación de la función integranda mediante polinomios de interpolación. Sea $f(x)$ una función continua en el intervalo $[a, b]$, dividimos este intervalo en n subintervalos iguales de longitud $h = (b - a)/n$. Los puntos de división son $x_i = a + ih$, donde $i = 0, 1, \dots, n$ (Burden & Faires, 2011). La integral se aproxima como:

$$\int_a^b f(x)dx \approx \sum_{i=0}^n w_i f(x_i)$$

donde w_i son los coeficientes de peso. Para obtener estos coeficientes, se integra el polinomio de interpolación de Lagrange:

$$P_n(x) = \sum_{i=0}^n f(x_i) L_i(x)$$

donde $L_i(x)$ son los polinomios de Lagrange:

$$L_i(x) = \prod_{j=0, j \neq i}^n \frac{x - x_j}{x_i - x_j}$$

Propiedades principales:

1. **Exactitud polinomial:** Las fórmulas de Newton-Cotes son exactas para polinomios de grado menor o igual a n para $n + 1$ puntos equidistantes (Davis & Rabinowitz, 1984). Simetría: Los coeficientes de peso son simétricos respecto al punto medio del intervalo. Convergencia: Para funciones suficientemente suaves, las fórmulas convergen a la integral exacta cuando $n \rightarrow \infty$.
2. **Análisis de error:** El error de las fórmulas de Newton-Cotes se puede expresar como:

$$E_n = \int_a^b f(x)dx - \sum_{i=0}^n w_i f(x_i) = K_n h^{p+1} f^{(p)}(\xi)$$

donde p es el orden de precisión, K_n es una constante que depende de n , y $\xi \in (a, b)$ (Chapra & Canale, 2015).

4.2.2 Cuadratura de Gauss

Derivación matemática:

La cuadratura de Gauss se basa en la elección óptima de los puntos de evaluación y los pesos para maximizar la precisión. Se busca aproximar:

$$\int_a^b w(x)f(x)dx \approx \sum_{i=1}^n w_i f(x_i)$$

donde $w(x)$ es una función de peso. Los puntos x_i son las raíces de los polinomios ortogonales respecto a la función de peso $w(x)$. Para el caso estándar (intervalo $[-1,1]$ y $w(x) = 1$), se utilizan los polinomios de Legendre (Burden & Faires, 2011). Los pesos w_i se calculan como:

$$w_i = \frac{2}{(1 - x_i^2)[P'_n(x_i)]^2}$$

donde $P_n(x)$ es el polinomio de Legendre de grado n .

Propiedades principales:

1. **Precisión óptima:** La cuadratura de Gauss de n puntos es exacta para polinomios de grado hasta $2n-1$ (Davis & Rabinowitz, 1984). Eficiencia: Requiere menos puntos de

evaluación que las fórmulas de Newton-Cotes para alcanzar la misma precisión.

2. **Adaptabilidad:** Puede adaptarse a diferentes funciones de peso y dominios de integración.
3. **Análisis de error:** El error de la cuadratura de Gauss se puede expresar como:

$$E_n = \frac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^3} f^{(2n)}(\xi)$$

donde $\xi \in (-1, 1)$ para el caso estándar (Chapra & Canale, 2015). Para integrales en un intervalo general $[a, b]$, se aplica un cambio de variable y el error se ajusta en consecuencia.

4.3 Implementación computacional de los métodos numéricos

La implementación computacional de los métodos numéricos es un paso crucial en el análisis comparativo de las técnicas de integración numérica. Esta sección se centra en la traducción de los conceptos matemáticos discutidos anteriormente en código ejecutable, permitiendo así la aplicación práctica y la evaluación de estos métodos. En esta implementación, nos enfocamos en tres métodos de Newton-Cotes: la regla del trapecio, la regla de Simpson 1/3, la regla de Simpson 3/8 y cuadratura de Gauss. Cada uno de estos métodos ofrece diferentes niveles de precisión y eficiencia en el cálculo de integrales definidas. Para llevar a cabo esta implementación, hemos elegido el lenguaje de programación Python, conocido por su versatilidad y amplia gama de bibliotecas científicas. En las siguientes subsecciones, describiremos en detalle el entorno de desarrollo utilizado y presentaremos la implementación específica de cada método, incluyendo pseudocódigo y fragmentos de código relevantes. Esta aproximación nos permitirá no solo calcular integrales numéricamente, sino también visualizar y comparar los resultados de los diferentes métodos.

4.3.1 Descripción del entorno de desarrollo

Lenguaje de programación utilizado: Python

Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Se eligió Python para este proyecto debido a su simplicidad, legibilidad y la amplia gama de bibliotecas científicas disponibles. Bibliotecas y herramientas empleadas:

- **SymPy:** Una biblioteca para matemáticas simbólicas en Python. Se utiliza para manejar expresiones matemáticas y realizar cálculos simbólicos.
- **NumPy:** Una biblioteca fundamental para computación científica en Python. Se utiliza para operaciones numéricas eficientes y manejo de arrays.
- **Matplotlib:** Una biblioteca de trazado para Python. Se utiliza para crear visualizaciones gráficas de las funciones y áreas bajo la curva.

Estas bibliotecas se importan al principio de cada script:

```
1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
```

4.3.2 Implementación de las fórmulas de Newton-Cotes

Se implementan tres métodos de Newton-Cotes: la regla del trapecio, la regla de Simpson 1/3 y la regla de Simpson 3/8.

a) Regla del Trapecio

Pseudocódigo:

1. Definir la función `trapezoidal_method(f, a, b, n)`
2. Calcular $h = (b - a) / n$
3. Inicializar integral con los valores en los extremos
4. Sumar los valores intermedios
5. Multiplicar el resultado por $h/2$
6. Devolver el resultado

Implementación en Python

```

1 def trapezoidal_method(f, a, b, n):
2     h = (b - a) / n
3     x = sp.Symbol('x')
4     integral = 0.5 * (f.subs(x, a) + f.subs(x, b)) # Suma de los
extremos
5     for i in range(1, n):
6         x_val = a + i * h
7         integral += f.subs(x, x_val) # Suma de los valores en el
interior
8     integral *= h
9     return integral

```

b) Regla de Simpson 1/3

Pseudocódigo:

1. Definir la función `simpson_one_third_method(f, a, b, n)`.
2. Verificar que `n` sea par.
3. Calcular $h = (b - a) / n$.
4. Inicializar `integral` con los valores en los extremos.
5. Sumar los valores intermedios con coeficientes alternados de 4 y 2.
6. Multiplicar el resultado por $h/3$.
7. Devolver el resultado.

Implementación en Python:

```

1 def simpson_one_third_method(f, a, b, n):
2     h = (b - a) / n
3     x = sp.Symbol('x')
4     integral = 0
5
6     for i in range(n + 1):
7         x_val = a + i * h
8         if i == 0 or i == n:
9             coefficient = 1

```

```

10         elif i % 2 == 1:
11             coefficient = 4
12         else:
13             coefficient = 2
14             integral += coefficient * f.subs(x, x_val)
15
16     integral *= h / 3
17     return integral

```

c) Regla de Simpson 3/8

Pseudocódigo:

1. Definir la función `simpson_three_eighth_method(f, a, b, n)`
2. Verificar que `n` sea divisible por 3
3. Calcular $h = (b - a) / n$
4. Inicializar `integral` con los valores en los extremos
5. Sumar los valores intermedios con coeficientes según el patrón 3, 3, 2
6. Multiplicar el resultado por $3h/8$
7. Devolver el resultado

Implementación en Python:

```

1 def simpson_three_eighth_method(f, a, b, n):
2     if n % 3 != 0:
3         print("El número de subintervalos debe ser divisible por 3
4         para el método de Simpson 3/8.")
5         return None
6
7     h = (b - a) / n
8     x = sp.Symbol('x')
9     integral = f.subs(x, a) + f.subs(x, b) # Suma de los extremos
10
11     for i in range(1, n, 3):
12         x_val1 = a + i * h

```

```

12     x_val2 = a + (i + 1) * h
13     integral += 3 * (f.subs(x, x_val1) + f.subs(x, x_val2)) #
Suma de los valores en el interior
14
15     for i in range(3, n, 3):
16         x_val = a + i * h
17         integral += 2 * f.subs(x, x_val) # Suma de los valores en
el interior
18
19     integral *= 3 * h / 8
20     return integral

```

Estas implementaciones permiten calcular la integral definida de una función utilizando los métodos de Newton-Cotes(Ver Anexo A para script completo). Cada método tiene sus propias características y precisión, y la elección entre ellos dependerá de la función específica y el nivel de precisión requerido.

4.3.3 Implementación de la cuadratura de Gauss

La cuadratura de Gauss es un método de integración numérica que ofrece una alta precisión con un número relativamente bajo de puntos de evaluación. A diferencia de los métodos de Newton-Cotes, que utilizan puntos equidistantes, la cuadratura de Gauss selecciona los puntos de evaluación (nodos) y los pesos de manera óptima.

Pseudocódigo:

1. Definir la función `cuadratura_gauss(f, a, b, n)`
2. Obtener los puntos y pesos de Gauss-Legendre para n puntos
3. Transformar los puntos al intervalo $[a, b]$
4. Calcular la suma ponderada de los valores de la función en los puntos transformados
5. Devolver el resultado de la integral aproximada

Explicación del algoritmo:

El algoritmo de la cuadratura de Gauss se basa en aproximar la integral definida mediante

una suma ponderada de los valores de la función en puntos específicos. Los pasos clave son:

1. Obtención de los puntos y pesos de Gauss-Legendre: Estos se calculan utilizando los polinomios de Legendre y sus propiedades.
2. Transformación de los puntos: Los puntos obtenidos están en el intervalo $[-1, 1]$, por lo que se transforman al intervalo de integración $[a, b]$.
3. Cálculo de la suma ponderada: Se evalúa la función en los puntos transformados y se multiplica cada valor por su peso correspondiente.

Implementación en Python:

```

1 import sympy as sp
2 import numpy as np
3
4 def cuadratura_gauss_legendre(f, a, b, n):
5     # Obtener los puntos y pesos de Gauss-Legendre
6     [puntos, pesos] = np.polynomial.legendre.leggauss(n)
7
8     # Transformar los puntos a la escala [a, b]
9     puntos_transformados = 0.5 * (puntos + 1) * (b - a) + a
10    pesos_transformados = 0.5 * (b - a) * pesos
11
12    # Calcular la integral aproximada
13    x = sp.Symbol('x')
14    integral = sum(pesos_transformados * np.array([f.subs(x, pt) for pt
15    in puntos_transformados], dtype=float))
16
17    return integral

```

Este fragmento proporciona la implementación esencial del método de cuadratura de Gauss, que puede ser fácilmente integrado en un script más amplio (Ver Anexo A) para análisis y comparación con otros métodos de integración numérica.

4.4 Diseño de experimentos y evaluación de desempeño

El diseño de experimentos y la evaluación de desempeño son componentes cruciales en el análisis comparativo de los métodos numéricos para el cálculo de integrales definidas. Esta sección describe la metodología sistemática empleada para evaluar y comparar la eficacia de los métodos. El enfoque se basa en tres pilares fundamentales: la selección cuidadosa de funciones de prueba que representan diversos desafíos matemáticos, la definición de parámetros de evaluación claros y medibles, y el establecimiento de una metodología experimental rigurosa. Este proceso permite no solo cuantificar el rendimiento de cada método en términos de precisión y eficiencia computacional, sino también comprender sus fortalezas y limitaciones en diferentes escenarios. A través de estos experimentos, se busca proporcionar una visión completa y objetiva del comportamiento de estos métodos numéricos, ofreciendo así una guía práctica para la selección del método más apropiado según las características específicas del problema de integración en cuestión.

4.4.1 Selección de funciones de prueba

Para evaluar el rendimiento de los métodos numéricos implementados, se han seleccionado un conjunto diverso de funciones de prueba que representan diferentes desafíos para la integración numérica:

Funciones suaves:

1. $f(x) = x^2$
2. $f(x) = \sin(x)$
3. $f(x) = e^x$

Estas funciones son continuas y diferenciables en todo su dominio, lo que las hace ideales para evaluar el rendimiento básico de los métodos.

Funciones con oscilaciones:

1. $f(x) = \sin(10x)$

$$2. f(x) = \cos(5x) * e^{(-x)}$$

$$3. f(x) = \sin(x^2)$$

Estas funciones presentan oscilaciones rápidas, lo que puede desafiar la precisión de los métodos, especialmente con un número bajo de subintervalos.

Funciones con singularidades:

$$1. f(x) = 1/x \text{ (en un intervalo que incluye } x=0)$$

$$2. f(x) = \log(x) \text{ (en un intervalo que incluye } x=0)$$

$$3. f(x) = \sqrt{(|x|)}$$

Estas funciones tienen puntos donde no son diferenciables o tienden al infinito, lo que puede causar problemas de precisión en los métodos numéricos.

4.4.2 Precisión de los métodos

La precisión es un factor crucial en la evaluación de métodos numéricos para el cálculo de integrales definidas. En esta sección, se analiza la precisión de los métodos de Newton-Cotes (regla del trapecio, Simpson 1/3 y Simpson 3/8) y la cuadratura de Gauss. Se seleccionan diversas funciones representativas para evaluar el comportamiento de los métodos en diferentes escenarios, desde funciones polinómicas suaves hasta aquellas con singularidades.

Se presentan tablas comparativas que muestran el valor aproximado de la integral, el valor real (cuando es conocido), el error relativo y el tiempo de ejecución para cada método y función. El número de intervalos utilizados en cada método también se incluye, ya que puede afectar significativamente tanto la precisión como el tiempo de ejecución.

Estas tablas permiten visualizar fácilmente las diferencias en precisión y eficiencia entre los métodos para cada función específica. A través de ellas, se pueden observar pa-

trones de precisión, identificar los métodos más adecuados para ciertos tipos de funciones y comprender la relación entre la precisión y el costo computacional.

Cuadro para funciones suaves

Para evaluar la precisión y eficiencia de los métodos numéricos, se han analizado tres funciones suaves en el intervalo $[0, 2]$. Se aplican los métodos del trapecio, Simpson 1/3, Simpson 3/8 y cuadratura de Gauss con diferentes números de subintervalos: el mínimo requerido por cada método, 10 subintervalos (o 9 para Simpson 3/8), y 100 subintervalos (99 para Simpson 3/8). Los siguientes cuadro muestran los resultados obtenidos, incluyendo el valor aproximado de la integral, el valor real, el error relativo y el tiempo de ejecución para cada configuración.

Cuadro 4.1: Comparación de métodos numéricos para funciones suaves con el número mínimo de subintervalos

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = x^2$	1	Trapecio	4.00000000000000	2.66666666666667	50.000000 %	0.000000
	2	Simpson 1/3	2.66666666666667	2.66666666666667	0.000000 %	0.000000
	3	Simpson 3/8	2.66666666666667	2.66666666666667	0.000000 %	0.003337
	2	Gauss	2.66666666666667	2.66666666666667	0.000000 %	0.000000
$f(x) = \sin(x)$	1	Trapecio	0.909297426825682	1.41614683654714	35.790738 %	0.000000
	2	Simpson 1/3	1.42506045535242	1.41614683654714	0.629428 %	0.000000
	3	Simpson 3/8	1.42005513503121	1.41614683654714	0.275981 %	0.015613
	2	Gauss	1.4101569816653807	1.41614683654714	0.422968 %	0.009936
$f(x) = e^x$	1	Trapecio	8.38905609893065	6.38905609893065	31.303529 %	0.000957
	2	Simpson 1/3	6.42072780425561	6.38905609893065	0.495718 %	0.000000
	3	Simpson 3/8	6.40331547653605	6.38905609893065	0.223184 %	0.000000
	2	Gauss	6.368108205367115	6.38905609893065	0.327871 %	0.015639

Fuente: Creación propia

Cuadro 4.2: Comparación de métodos numéricos para funciones suaves con aproximadamente 10 subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = x^2$	10	Trapecio	2.68	2.66666666666667	0.500000 %	0.021000
	10	Simpson 1/3	2.66666666666667	2.66666666666667	0.000000 %	0.000000
	9	Simpson 3/8	2.66666666666667	2.66666666666667	0.000000 %	0.000000
	10	Gauss	2.66666666666666	2.66666666666667	0.000000 %	0.033581
$f(x) = \sin(x)$	10	Trapecio	1.41142319709888	1.41614683654714	0.333556 %	0.000000
	10	Simpson 1/3	1.41615948471454	1.41614683654714	0.000893 %	0.012534
	9	Simpson 3/8	1.41619051850974	1.41614683654714	0.003085 %	0.015613
	10	Gauss	1.4161468365471426	1.41614683654714	0.000000 %	0.000000
$f(x) = e^x$	10	Trapecio	6.41033876819961	6.38905609893065	0.333111 %	0.000957
	10	Simpson 1/3	6.38911262123539	6.38905609893065	0.000885 %	0.000000
	9	Simpson 3/8	6.38924859304734	6.38905609893065	0.003013 %	0.000000
	10	Gauss	6.38905609893065	6.38905609893065	0.000000 %	0.000000

Fuente: Creación propia

Cuadro 4.3: Comparación de métodos numéricos para funciones suaves con aproximadamente 100 subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = x^2$	100	Trapecio	2.66680000000000	2.66666666666667	0.005000 %	0.038992
	100	Simpson 1/3	2.66666666666667	2.66666666666667	0.000000 %	0.048015
	99	Simpson 3/8	2.66666666666667	2.66666666666667	0.000000 %	0.033242
	100	Gauss	2.666666666666603	2.66666666666667	0.000000 %	0.053193
$f(x) = \sin(x)$	100	Trapecio	1.41609963133789	1.41614683654714	0.003333 %	0.034044
	100	Simpson 1/3	1.41614683780600	1.41614683654714	0.000000 %	0.050047
	99	Simpson 3/8	1.41614683949590	1.41614683654714	0.000000 %	0.058371
	100	Gauss	1.4161468365471437	1.41614683654714	0.000000 %	0.092734
$f(x) = e^x$	100	Trapecio	6.38926906604750	6.38905609893065	0.003333 %	0.030589
	100	Simpson 1/3	6.38905610460954	6.38905609893065	0.000000 %	0.054459
	99	Simpson 3/8	6.38905611223163	6.38905609893065	0.000000 %	0.033273
	100	Gauss	6.38905609893065	6.38905609893065	0.000000 %	0.037023

Fuente: Creación propia

En los cuadros 4.1, 4.2 y 4.3, se puede observar cómo la precisión y el tiempo de ejecución varían con el aumento de subintervalos para cada método y función. Notablemente, para $f(x) = x^2$, los métodos de Simpson y Gauss logran precisión perfecta incluso con pocos subintervalos. Para funciones más complejas como $\sin(x)$ y e^x , vemos una mejora significativa en la precisión al aumentar el número de subintervalos, especialmente para el método del trapecio.

Es interesante notar que el método de cuadratura de Gauss logra una precisión excelente en todos los casos, incluso con pocos puntos de evaluación. Sin embargo, su tiempo de ejecución tiende a ser mayor que el de los otros métodos, especialmente cuando se aumenta el número de puntos.

Tabla para funciones con oscilaciones

Para evaluar el rendimiento de los métodos numéricos en funciones con oscilaciones, se han seleccionado tres funciones características, todas evaluadas en el intervalo $[0, 2]$. Se aplican los métodos del trapecio, Simpson 1/3, Simpson 3/8 y cuadratura de Gauss utilizando el número mínimo de subintervalos requerido por cada método, así como aproximadamente 10 y 100 subintervalos. Para el método de Simpson 3/8, que requiere un número de subintervalos múltiplo de 3, utilizamos 9 y 99 subintervalos en las pruebas correspondientes. Esta selección permite examinar cómo la precisión y el tiempo de ejecución varían con el aumento de subintervalos para funciones que presentan cambios rápidos y frecuentes.

Cuadro 4.4: Comparación de métodos numéricos para funciones oscilantes con el número mínimo de subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = \sin(10x)$	1	Trapecio	0.912945250727628	0.0591917938186608	1442.351 %	0.000000
	2	Simpson 1/3	0.421046397609950	0.0591917938186608	611.3256 %	0.000000
	3	Simpson 3/8	1.02931338654311	0.0591917938186608	1638.946 %	0.000000
	2	Gauss	0.9497514065728347	0.0591917938186608	1704.532 %	0.000614
$f(x) = \cos(5x)e^{-x}$	1	Trapecio	-0.886444016956661	0.0286703741307228	2991.846 %	0.000000
	2	Simpson 1/3	-0.434619320678462	0.0286703741307228	1615.918 %	0.016698
	3	Simpson 3/8	0.0269434291220974	0.0286703741307228	6.023 %	0.000368
	2	Gauss	0.3450612850719687	0.0286703741307228	1303.546 %	0.000000
$f(x) = \sin(x^2)$	1	Trapecio	-0.756802495307928	0.8047764893437562	194.039 %	0.000957
	2	Simpson 1/3	0.869693814641219	0.8047764893437562	8.067 %	0.000000
	3	Simpson 3/8	0.867258427168662	0.8047764893437562	7.764 %	0.000000
	2	Gauss	0.7856999464919483	0.8047764893437562	2.3704 %	0.000000

Fuente: Creación propia

Cuadro 4.5: Comparación de métodos numéricos para funciones oscilantes con aproximadamente 10 subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = \sin(10x)$	10	Trapecio	0.0380066137348700	0.0591917938186608	35.791 %	0.017170
	10	Simpson 1/3	0.0687351993779909	0.0591917938186608	16.122 %	0.003561
	9	Simpson 3/8	-0.0904188259437582	0.0591917938186608	252.755 %	0.009014
	10	Gauss	0.059212227306949146	0.0591917938186608	0.0345 %	0.016505
$f(x) = \cos(5x)e^{-x}$	10	Trapecio	0.0338177198710158	0.0286703741307228	17.953 %	0.000000
	10	Simpson 1/3	0.0277047980444085	0.0286703741307228	3.367 %	0.014287
	9	Simpson 3/8	0.0241687637836937	0.0286703741307228	15.701 %	0.000000
	10	Gauss	0.028670374170156644	0.0286703741307228	0.000000 %	0.00601
$f(x) = \sin(x^2)$	10	Trapecio	0.795924732850834	0.8047764893437562	1.099 %	0.013531
	10	Simpson 1/3	0.805376152475580	0.8047764893437562	0.0745 %	0.009444
	9	Simpson 3/8	0.807182766422415	0.8047764893437562	0.299 %	0.000000
	10	Gauss	0.8047764893239352	0.8047764893437562	0.000000 %	0.000000

Fuente: Creación propia

Cuadro 4.6: Comparación de métodos numéricos para funciones oscilantes con aproximadamente 100 subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = \sin(10x)$	100	Trapecio	0.0589943561765467	0.0591917938186608	0.3335 %	0.049065
	100	Simpson 1/3	0.0591923224839781	0.0591917938186608	0.000893 %	0.062975
	99	Simpson 3/8	0.0591930383035651	0.0591917938186608	0.0021 %	0.055795
	100	Gauss	0.059191793818656566	0.0591917938186608	0.000000 %	0.086828
$f(x) = \cos(5x)e^{-x}$	100	Trapecio	0.0287197836572929	0.0286703741307228	0.1723 %	0.098160
	100	Simpson 1/3	0.0286702935429142	0.0286703741307228	0.000281 %	0.110237
	99	Simpson 3/8	0.0286701850239361	0.0286703741307228	0.000660 %	0.215236
	100	Gauss	0.028670374130718743	0.0286703741307228	0.000000 %	0.129755
$f(x) = \sin(x^2)$	100	Trapecio	0.804689323525294	0.8047764893437562	0.010831 %	0.055674
	100	Simpson 1/3	0.804776542736671	0.8047764893437562	0.000007 %	0.053144
	99	Simpson 3/8	0.804776614558907	0.8047764893437562	0.000016 %	0.060372
	100	Gauss	0.8047764893437633	0.8047764893437562	0.000000 %	0.061287

Fuente: Creación propia

En los Cuadros 4.4, 4.5 y 4.6 nos permiten observar cómo la precisión y el tiempo de ejecución varían con el aumento de subintervalos para cada método y función con oscilaciones. Notablemente, para todas las funciones analizadas, los métodos muestran un rendimiento pobre con pocos subintervalos, lo que refleja la complejidad de integrar funciones oscilantes. Vemos una mejora dramática en la precisión al aumentar el número de subintervalos, especialmente al pasar de 10 a 100 subintervalos.

Es interesante notar que el método de cuadratura de Gauss logra una precisión excelente en todos los casos cuando se utilizan 100 puntos, superando a los otros métodos en términos de error relativo. Sin embargo, su rendimiento con pocos puntos de evaluación es inconsistente, siendo a veces el mejor (como en $f(x) = \sin(x^2)$ con 2 puntos) y otras veces el peor (como en $f(x) = \sin(10x)$ con 2 puntos). El tiempo de ejecución de Gauss tiende a ser mayor que el de los otros métodos, especialmente cuando se aumenta el número de puntos. Los métodos de Simpson muestran una mejora consistente con el aumento de subintervalos, pero su rendimiento varía según la función. Por ejemplo, Simpson 3/8 muestra un rendimiento sorprendentemente bueno para $f(x) = \cos(5x) * e^{(-x)}$ incluso con pocos subintervalos. El método del trapecio, aunque menos preciso con pocos subintervalos, muestra una mejora significativa y tiempos de ejecución competitivos con 100 subintervalos.

Tabla para funciones con singularidades

Para evaluar el rendimiento de los métodos numéricos en funciones con singularidades, se han seleccionado tres funciones características, para las dos primeras funciones

son evaluadas en el intervalo $[0.1, 2]$ y $[0, 2]$ para la tercera. Se eligió 0.1 como extremo inferior para $1/x$ y $\log(x)$ para evitar la singularidad en $x = 0$, donde estas funciones no están definidas. Aplicamos los métodos del trapecio, Simpson 1/3, Simpson 3/8 y cuadratura de Gauss utilizando el número mínimo de subintervalos requerido por cada método, así como aproximadamente 10 y 100 subintervalos. Para el método de Simpson 3/8, que requiere un número de subintervalos múltiplo de 3, utilizamos 9 y 99 subintervalos en las pruebas correspondientes.

Cuadro 4.7: Comparación de métodos numéricos para funciones con singularidades con el número mínimo de subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = \frac{1}{x}$	1	Trapecio	9.97500000000000	2.99573227355399	232.973680 %	0.000000
	2	Simpson 1/3	4.53134920634921	2.99573227355399	51.260153 %	0.012468
	3	Simpson 3/8	3.98668237250554	2.99573227355399	33.078727 %	0.000000
	2	Gauss	2.4885654885654884	2.99573227355399	16.929643 %	0.016309
$f(x) = \log(x)$	1	Trapecio	-1.52896601681240	-0.283447129580705	439.418416 %	0.000000
	2	Simpson 1/3	-0.447854464322851	-0.283447129580705	58.002822 %	0.000000
	3	Simpson 3/8	-0.380659927527051	-0.283447129580705	34.296625 %	0.000000
	2	Gauss	-0.21000926585486596	-0.283447129580705	25.908840 %	0.017307
$f(x) = \sqrt{ x }$	1	Trapecio	1.41421356237310	1.88561808316413	25.000000 %	0.000000
	2	Simpson 1/3	1.80473785412436	1.88561808316413	4.289322 %	0.000000
	3	Simpson 3/8	1.83195123007351	1.88561808316413	2.846115 %	0.000000
	2	Gauss	1.9060412277428451	1.88561808316413	1.083101 %	0.000000

Fuente: Creación propia

Cuadro 4.8: Comparación de métodos numéricos para funciones con singularidades con aproximadamente 10 subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = \frac{1}{x}$	10	Trapecio	3.24231971581583	2.99573227355399	8.231291 %	0.014808
	10	Simpson 1/3	3.06657740578087	2.99573227355399	2.364869 %	0.000000
	9	Simpson 3/8	3.12148441918959	2.99573227355399	4.197710 %	0.002317
	10	Gauss	2.9953110431836945	2.99573227355399	0.014061 %	0.000000
$f(x) = \log(x)$	10	Trapecio	-0.309801452599567	-0.283447129580705	9.297791 %	0.000000
	10	Simpson 1/3	-0.287418627791148	-0.283447129580705	1.401143 %	0.008641
	9	Simpson 3/8	-0.291467552072042	-0.283447129580705	2.829601 %	0.012001
	10	Gauss	-0.2834370834698764	-0.283447129580705	0.003544 %	0.016930
$f(x) = \sqrt{ x }$	10	Trapecio	1.86820253823181	1.88561808316413	0.923599 %	0.000000
	10	Simpson 1/3	1.87835729320271	1.88561808316413	0.385062 %	0.016421
	9	Simpson 3/8	1.87526529458739	1.88561808316413	0.549040 %	0.000432
	10	Gauss	1.885870877430057	1.88561808316413	0.013406 %	0.020332

Fuente: Creación propia

como se puede observar en los cuadro 4.7, 4.8 y 4.9 las funciones con singularidades presentan desafíos significativos para los métodos de integración numérica, especialmente con pocos subintervalos. La precisión mejora notablemente al aumentar el número de

Cuadro 4.9: Comparación de métodos numéricos para funciones con singularidades con aproximadamente 100 subintervalos.

Función	Número de intervalo	Método	Valor Aproximado	Valor Real	Error Relativo	Tiempo de Ejecución (s)
$f(x) = \frac{1}{x}$	100	Trapecio	2.99872240607547	2.99573227355399	0.099813 %	0.230670
	100	Simpson 1/3	2.99577245843891	2.99573227355399	0.001341 %	0.032889
	99	Simpson 3/8	2.99582032826903	2.99573227355399	0.002939 %	0.039586
	100	Gauss	2.9957322735539575	2.99573227355399	0.000000 %	0.047213
$f(x) = \log(x)$	100	Trapecio	-0.28373256292733	-0.283447129580705	0.100701 %	0.063611
	100	Simpson 1/3	-0.283448509875571	-0.283447129580705	0.000487 %	0.085901
	99	Simpson 3/8	-0.283450227940242	-0.283447129580705	0.001093 %	0.064681
	100	Gauss	-0.2834471295806992	-0.283447129580705	0.000000 %	0.098592
$f(x) = \sqrt{ x }$	100	Trapecio	1.88504187722483	1.88561808316413	0.030558 %	0.034588
	100	Simpson 1/3	1.88538845843954	1.88561808316413	0.012178 %	0.052747
	99	Simpson 3/8	1.88533425804910	1.88561808316413	0.015052 %	0.050972
	100	Gauss	1.885618370057271	1.88561808316413	0.000015 %	0.046553

Fuente: Creación propia

subintervalos, con el método de Gauss mostrando generalmente la mejor precisión. Sin embargo, todos los métodos muestran una mejora sustancial de precisión al pasar de 10 a 100 subintervalos, lo que subraya la importancia de usar un número adecuado de puntos de evaluación para funciones con comportamiento singular.

4.5 Discusión de resultados

4.5.1 Ventajas y desventajas de cada método

Los métodos de Newton-Cotes y la cuadratura de Gauss presentan diferentes fortalezas y debilidades que se han evidenciado a través de los experimentos realizados.

La regla del trapecio, el método más simple de Newton-Cotes, destaca por su facilidad de implementación y bajo costo computacional. Sin embargo, su precisión es limitada, especialmente para funciones con curvaturas pronunciadas o oscilaciones. Es particularmente útil cuando se requiere una estimación rápida y aproximada de la integral.

Las reglas de Simpson (1/3 y 3/8) ofrecen una mejora significativa en precisión respecto a la regla del trapecio, manteniendo una complejidad computacional relativamente baja. Son especialmente efectivas para funciones suaves y moderadamente oscilantes. No obstante, pueden presentar dificultades con funciones que tienen singularidades o cambios abruptos.

La cuadratura de Gauss destaca por su alta precisión, incluso con un número re-

ducido de puntos de evaluación. Es particularmente eficaz para funciones suaves y puede manejar bien ciertas singularidades. Sin embargo, su implementación es más compleja y el cálculo de los nodos y pesos puede ser computacionalmente costoso para un gran número de puntos.

4.5.2 Recomendaciones de uso según el tipo de función

Para funciones suaves y bien comportadas, todos los métodos proporcionan buenos resultados, pero la cuadratura de Gauss y las reglas de Simpson ofrecen la mejor relación entre precisión y eficiencia computacional.

En el caso de funciones con oscilaciones moderadas, las reglas de Simpson son una excelente opción, ofreciendo un buen equilibrio entre precisión y facilidad de implementación.

Para funciones con oscilaciones rápidas, la cuadratura de Gauss tiende a proporcionar los mejores resultados, especialmente si se aumenta el número de puntos de evaluación.

En presencia de singularidades, la elección del método depende de la naturaleza de la singularidad. La cuadratura de Gauss puede manejar bien ciertas singularidades, mientras que para otras puede ser necesario recurrir a técnicas adaptativas o de subdivisión del intervalo.

CAPÍTULO V

CONCLUSIONES

5.1 Introducción

Este capítulo presenta las conclusiones derivadas del análisis comparativo de los métodos numéricos para el cálculo de integrales definidas, centrándose en las fórmulas de Newton-Cotes y la cuadratura de Gauss. Se abordan los objetivos planteados inicialmente, destacando los hallazgos clave y las implicaciones prácticas de este estudio.

5.2 Conclusiones

El análisis comparativo de los métodos numéricos para el cálculo de integrales definidas, centrado en las fórmulas de Newton-Cotes y la cuadratura de Gauss, ha proporcionado valiosas perspectivas sobre su rendimiento, precisión y aplicabilidad. A continuación, se presentan las principales conclusiones derivadas de este estudio:

1. El examen de los fundamentos teóricos ha revelado la base matemática sólida de estos métodos. Las fórmulas de Newton-Cotes, basadas en la interpolación polinómica, ofrecen una aproximación intuitiva, mientras que la cuadratura de Gauss, con su optimización de puntos de evaluación, demuestra una eficiencia notable. El análisis de error ha puesto de manifiesto que la cuadratura de Gauss generalmente presenta una convergencia más rápida que los métodos de Newton-Cotes para funciones suaves.
2. La implementación computacional en Python y los experimentos realizados han permitido una evaluación práctica y exhaustiva del rendimiento de cada método en diversos escenarios. Esta fase experimental ha sido crucial para comprender el comportamiento real de estos métodos más allá de sus fundamentos teóricos.
3. La comparación de precisión y velocidad de convergencia en diferentes tipos de funciones ha arrojado resultados significativos:
 - Para funciones suaves, todos los métodos convergen, pero la cuadratura de Gauss destaca por su precisión superior con un menor número de puntos de

evaluación.

- En el caso de funciones oscilantes, las reglas de Simpson y la cuadratura de Gauss demuestran un rendimiento notablemente superior a la regla del trapecio, especialmente al aumentar el número de puntos.
- Frente a funciones con singularidades, la cuadratura de Gauss suele ofrecer los mejores resultados, aunque todos los métodos requieren un número elevado de puntos para alcanzar una precisión aceptable.

Este estudio ha demostrado que la selección del método de integración numérica más adecuado debe basarse en un análisis cuidadoso de las características específicas de la función a integrar, así como en los requisitos de precisión y eficiencia computacional del problema en cuestión. La cuadratura de Gauss se distingue por su alta precisión general, pero los métodos de Newton-Cotes, particularmente las reglas de Simpson, ofrecen un equilibrio favorable entre precisión y facilidad de implementación para una amplia gama de aplicaciones prácticas.

5.3 Líneas futuras de investigación

Este estudio abre varias líneas potenciales para futuras investigaciones:

1. Exploración de métodos adaptativos que combinen diferentes técnicas de integración numérica según las características locales de la función.
2. Investigación sobre la aplicación de estos métodos en integrales multidimensionales y su eficacia en problemas de mayor complejidad.
3. Desarrollo de algoritmos optimizados para la implementación eficiente de la cuadratura de Gauss en problemas de gran escala.
4. Estudio comparativo de estos métodos clásicos con técnicas más recientes, como los métodos de cuadratura basados en wavelets o redes neuronales.
5. Análisis del impacto de la aritmética de precisión finita en la estabilidad y precisión de estos métodos en implementaciones computacionales reales.

Referencias bibliográficas.

- Burden, R. L., & Faires, J. D. (2011). Numerical Analysis (9th ed.). Brooks/Cole, Cengage Learning.
- Chapra, S. C., & Canale, R. P. (2015). Numerical Methods for Engineers (7th ed.). McGraw-Hill Education.
- Cools, R. (1997). Constructing cubature formulae: the science behind the art. *Acta Numerica*, 6, 1-54. <https://doi.org/10.1017/S0962492900002701>
- Davis, P. J., & Rabinowitz, P. (1984). Methods of Numerical Integration (2nd ed.). Academic Press.
- Fernández, C. (2016). Cálculo Integral: Teoría y Problemas. Editorial Universidad de Granada.
- Gander, W., & Gautschi, W. (2000). Adaptive quadrature—revisited. *BIT Numerical Mathematics*, 40(1), 84-101. <https://doi.org/10.1023/A:1022318402393>
- Gutiérrez, J. M. (2017). Numerical Analysis: A Comprehensive Introduction. Princeton University Press.
- Hernández Sampieri, R., Fernández Collado, C., & Baptista Lucio, P. (2014). Metodología de la investigación (6ª ed.). McGraw-Hill.
- Larson, R., & Edwards, B. H. (2016). Calculus (10th ed.). Cengage Learning.
- Pérez, J. M. (2018). Historia del Cálculo Integral. Real Academia de Ciencias Exactas, Físicas y Naturales.

Sánchez, F. J. (2015). *Numerical Integration: Advanced Methods and Applications*. Wiley.

Sweldens, W., & Piessens, R. (1994). Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions. *SIAM Journal on Numerical Analysis*, 31(4), 1240-1264. <https://doi.org/10.1137/0731064>

ANEXO

Entorno de programación de los métodos de integración

Método del Trapecio

```

1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import time
5
6 def trapezoidal_method(f, a, b, n):
7     h = (b - a) / n
8     x = sp.Symbol('x')
9     integral = 0.5 * (f.subs(x, a) + f.subs(x, b)) # Suma de los
    extremos
10     for i in range(1, n):
11         x_val = a + i * h
12         integral += f.subs(x, x_val) # Suma de los valores en el
    interior
13     integral *= h
14     return integral
15
16 def exact_integral(f, a, b):
17     x = sp.Symbol('x')
18     integral = sp.integrate(f, (x, a, b))
19     return integral
20
21 # Graficar la función y el área bajo la curva
22 def plot_function_and_area(f, a, b, n):
23     x_vals = np.linspace(a, b, 400)
24     y_vals = [f.subs('x', x) for x in x_vals]
25
26     plt.plot(x_vals, y_vals, label=f'Función: {f}')
27
28     # Calcular la integral aproximada
29     start_time = time.time()
30     integral_approx = trapezoidal_method(f, a, b, n)
31     end_time = time.time()
32

```

```

33     # Crear puntos para graficar el área bajo la curva
34     x_area = np.linspace(a, b, n+1)
35     y_area = np.array([f.subs('x', x) for x in x_area], dtype=float) #
Convertir a arreglo de NumPy
36
37     plt.fill_between(x_area, y_area, alpha=0.3, label='Área bajo la
curva')
38
39     plt.xlabel('x')
40     plt.ylabel('f(x)')
41     plt.title('Gráfica de la función y el área bajo la curva (Método del
Trapezio)')
42     plt.legend()
43     plt.grid(True)
44     plt.show()
45
46     # Mostrar el valor numérico del área encontrada
47     print(f'El área aproximada bajo la curva es: {integral_approx}')
48
49     # Calcular la integral exacta y el error relativo
50     integral_exact = exact_integral(f, a, b)
51     error_relativo = abs((integral_exact - integral_approx) /
integral_exact) * 100
52
53     # Mostrar el tiempo de ejecución y el error relativo
54     print(f'Tiempo de ejecución: {end_time - start_time:.6f} segundos')
55     print(f'Integral exacta: {integral_exact}')
56     print(f'Error relativo: {error_relativo:.6f}%')
57
58 # Ejemplo de uso
59 def get_function():
60     while True:
61         try:
62             expression = input('Ingrese la función en términos de x: ')
63             x = sp.Symbol('x')
64             f = sp.sympify(expression)

```

```

65         return f
66     except:
67         print("Por favor, ingrese una expresión válida.")
68
69 a = float(input('Ingrese el límite inferior de integración (a): '))
70 b = float(input('Ingrese el límite superior de integración (b): '))
71 n = int(input('Ingrese el número de subintervalos: '))
72
73 if n < 1:
74     print("El número de subintervalos debe ser al menos 1.")
75 else:
76     function = get_function()
77     plot_function_and_area(function, a, b, n)

```

Método de Simpson 1/3

```

1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import time
5
6 def simpson_one_third_method(f, a, b, n):
7     h = (b - a) / n
8     x = sp.Symbol('x')
9     integral = 0
10
11     for i in range(n + 1):
12         x_val = a + i * h
13         if i == 0 or i == n:
14             coefficient = 1
15         elif i % 2 == 1:
16             coefficient = 4
17         else:
18             coefficient = 2
19         integral += coefficient * f.subs(x, x_val)
20
21     integral *= h / 3

```

```

22     return integral
23
24 def exact_integral(f, a, b):
25     x = sp.Symbol('x')
26     integral = sp.integrate(f, (x, a, b))
27     return integral
28
29 # Graficar la función y el área bajo la curva
30 def plot_function_and_area(f, a, b, n):
31     x_vals = np.linspace(a, b, 400)
32     y_vals = [f.subs('x', x) for x in x_vals]
33
34     plt.plot(x_vals, y_vals, label=f'Función: {f}')
35
36     # Calcular la integral aproximada
37     start_time = time.time()
38     integral_approx = simpson_one_third_method(f, a, b, n)
39     end_time = time.time()
40
41     # Crear puntos para graficar el área bajo la curva
42     x_area = np.linspace(a, b, n+1)
43     y_area = np.array([f.subs('x', x).evalf() for x in x_area], dtype=
44 float) # Convertir a arreglo de NumPy
45
46     plt.fill_between(x_area, y_area, alpha=0.3, label='Área bajo la
47 curva')
48
49     plt.xlabel('x')
50     plt.ylabel('f(x)')
51     plt.title('Gráfica de la función y el área bajo la curva')
52     plt.legend()
53     plt.grid(True)
54     plt.show()
55
56     # Mostrar el valor numérico del área encontrada
57     print(f'El área aproximada bajo la curva es: {integral_approx}')

```

```

56
57     # Calcular la integral exacta y el error relativo
58     integral_exact = exact_integral(f, a, b)
59     error_relativo = abs((integral_exact - integral_approx) /
60                           integral_exact) * 100
61
62     # Mostrar el tiempo de ejecución y el error relativo
63     print(f'Tiempo de ejecución: {end_time - start_time:.6f} segundos')
64     print(f'Integral exacta: {integral_exact}')
65     print(f'Error relativo: {error_relativo:.6f}%')
66
67 # Ejemplo de uso
68 def get_function():
69     while True:
70         try:
71             expression = input('Ingrese la función en términos de x: ')
72             x = sp.Symbol('x')
73             f = sp.sympify(expression)
74             return f
75         except:
76             print("Por favor, ingrese una expresión válida.")
77
78 a = float(input('Ingrese el límite inferior de integración (a): '))
79 b = float(input('Ingrese el límite superior de integración (b): '))
80 n = int(input('Ingrese el número de subintervalos (debe ser par): '))
81
82 if n % 2 != 0:
83     print("El número de subintervalos debe ser par para el método de
84     Simpson 1/3.")
85 else:
86     function = get_function()
87     plot_function_and_area(function, a, b, n)

```

Método de Simpson 3/8

```

1 import sympy as sp
2 import numpy as np

```

```

3 import matplotlib.pyplot as plt
4 import time
5
6 def simpson_three_eighth_method(f, a, b, n):
7     if n % 3 != 0:
8         print("El número de subintervalos debe ser divisible por 3 para
          el método de Simpson 3/8.")
9         return None
10
11     h = (b - a) / n
12     x = sp.Symbol('x')
13     integral = f.subs(x, a) + f.subs(x, b)  # Suma de los extremos
14
15     for i in range(1, n, 3):
16         x_val1 = a + i * h
17         x_val2 = a + (i + 1) * h
18         integral += 3 * (f.subs(x, x_val1) + f.subs(x, x_val2))  # Suma
          de los valores en el interior
19
20     for i in range(3, n, 3):
21         x_val = a + i * h
22         integral += 2 * f.subs(x, x_val)  # Suma de los valores en el
          interior
23
24     integral *= 3 * h / 8
25     return integral
26
27 def exact_integral(f, a, b):
28     x = sp.Symbol('x')
29     integral = sp.integrate(f, (x, a, b))
30     return integral
31
32 # Graficar la función y el área bajo la curva
33 def plot_function_and_area(f, a, b, n):
34     x_vals = np.linspace(a, b, 400)
35     y_vals = [f.subs('x', x) for x in x_vals]

```

```

36
37 plt.plot(x_vals, y_vals, label=f'Función: {f}')
38
39 # Calcular la integral aproximada
40 start_time = time.time()
41 integral_approx = simpson_three_eighth_method(f, a, b, n)
42 end_time = time.time()
43
44 # Crear puntos para graficar el área bajo la curva
45 x_area = np.linspace(a, b, n+1)
46 y_area = np.array([f.subs('x', x).evalf() for x in x_area], dtype=
float) # Convertir a arreglo de NumPy
47
48 plt.fill_between(x_area, y_area, alpha=0.3, label='Área bajo la
curva')
49
50 plt.xlabel('x')
51 plt.ylabel('f(x)')
52 plt.title('Gráfica de la función y el área bajo la curva (Método de
Simpson 3/8)')
53 plt.legend()
54 plt.grid(True)
55 plt.show()
56
57 # Mostrar el valor numérico del área encontrada
58 print(f'El área aproximada bajo la curva es: {integral_approx}')
59
60 # Calcular la integral exacta y el error relativo
61 integral_exact = exact_integral(f, a, b)
62 error_relativo = abs((integral_exact - integral_approx) /
integral_exact) * 100
63
64 # Mostrar el tiempo de ejecución y el error relativo
65 print(f'Tiempo de ejecución: {end_time - start_time:.6f} segundos')
66 print(f'Integral exacta: {integral_exact}')
67 print(f'Error relativo: {error_relativo:.6f}%')

```

```

68
69 # Ejemplo de uso
70 def get_function():
71     while True:
72         try:
73             expression = input('Ingrese la función en términos de x: ')
74             x = sp.Symbol('x')
75             f = sp.sympify(expression)
76             return f
77         except:
78             print("Por favor, ingrese una expresión válida.")
79
80 a = float(input('Ingrese el límite inferior de integración (a): '))
81 b = float(input('Ingrese el límite superior de integración (b): '))
82 n = int(input('Ingrese el número de subintervalos (debe ser divisible
    por 3): '))
83
84 if n < 3 or n % 3 != 0:
85     print("El número de subintervalos debe ser al menos 3 y divisible
    por 3 para el método de Simpson 3/8.")
86 else:
87     function = get_function()
88     plot_function_and_area(function, a, b, n)

```

Cuadranturas de Gauss

```

1 import sympy as sp
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import time
5
6 def cuadratura_gauss_legendre(f, a, b, n):
7     # Obtener los puntos y pesos de Gauss-Legendre
8     [puntos, pesos] = np.polynomial.legendre.leggauss(n)
9
10    # Transformar los puntos a la escala [a, b]
11    puntos_transformados = 0.5 * (puntos + 1) * (b - a) + a

```

```

12 pesos_transformados = 0.5 * (b - a) * pesos
13
14 # Calcular la integral aproximada
15 x = sp.Symbol('x')
16 integral = sum(pesos_transformados * np.array([f.subs(x, pt) for pt
17 in puntos_transformados], dtype=float))
18
19 return integral
20
21 def integral_exacta(f, a, b):
22     x = sp.Symbol('x')
23     integral = sp.integrate(f, (x, a, b))
24     return integral
25
26 # Graficar la función y el área bajo la curva
27 def graficar_funcion_y_area(f, a, b, n):
28     x_vals = np.linspace(a, b, 400)
29     y_vals = [f.subs('x', x) for x in x_vals]
30
31     plt.plot(x_vals, y_vals, label=f'Función: {f}')
32
33     # Calcular la integral aproximada
34     start_time = time.time()
35     integral_aprox = cuadratura_gauss_legendre(f, a, b, n)
36     end_time = time.time()
37
38     # Crear puntos para graficar el área bajo la curva
39     x_area = np.linspace(a, b, n)
40     y_area = np.array([f.subs('x', x) for x in x_area], dtype=float) #
41     Convertir a arreglo de NumPy
42
43     plt.fill_between(x_area, y_area, alpha=0.3, label='Área bajo la
44     curva')
45
46     plt.xlabel('x')
47     plt.ylabel('f(x)')

```

```

45 plt.title('Gráfica de la función y el área bajo la curva (Cuadratura
    de Gauss)')
46 plt.legend()
47 plt.grid(True)
48 plt.show()
49
50 # Mostrar el valor numérico del área encontrada
51 print(f'El área aproximada bajo la curva es: {integral_aprox}')
52
53 # Calcular la integral exacta y el error relativo
54 integral_exacta_valor = integral_exacta(f, a, b)
55 error_relativo = abs((integral_exacta_valor - integral_aprox) /
    integral_exacta_valor) * 100
56
57 # Mostrar el tiempo de ejecución y el error relativo
58 print(f'Tiempo de ejecución: {end_time - start_time:.6f} segundos')
59 print(f'Integral exacta: {integral_exacta_valor}')
60 print(f'Error relativo: {error_relativo:.6f}%')
61
62 # Ejemplo de uso
63 def obtener_funcion():
64     while True:
65         try:
66             expresion = input('Ingrese la función en términos de x: ')
67             x = sp.Symbol('x')
68             f = sp.sympify(expresion)
69             return f
70         except:
71             print("Por favor, ingrese una expresión válida.")
72
73 def obtener_float(mensaje):
74     while True:
75         try:
76             return float(input(mensaje))
77         except ValueError:
78             print("Por favor, ingrese un número válido.")

```

```
79
80 def obtener_int(mensaje):
81     while True:
82         try:
83             return int(input(mensaje))
84         except ValueError:
85             print("Por favor, ingrese un número entero válido.")
86
87 a = obtener_float('Ingrese el límite inferior de integración (a): ')
88 b = obtener_float('Ingrese el límite superior de integración (b): ')
89 n = obtener_int('Ingrese el número de puntos de Gauss: ')
90
91 if n < 1:
92     print("El número de puntos de Gauss debe ser al menos 1.")
93 else:
94     funcion = obtener_funcion()
95     graficar_funcion_y_area(funcion, a, b, n)
```