

Métodos Numéricos

Grupo: 4F21 **Semestre:** 4to

Actividad:
T1 - E2 - Problemario

Integrantes
Diego Alonso Coronel Vargas
Brandon García Ordaz
Oscar Aaron Delgadillo Fernández

En los cálculos numéricos, la precisión de los resultados depende de diversos factores, como la forma en que los números son representados y las operaciones realizadas sobre ellos. Los errores numéricos son inevitables y pueden clasificarse en diferentes tipos según su origen.

Clasificación de errores

Error de Redondeo:

Ocurre cuando un número es aproximado debido a la limitación de la cantidad de dígitos con los que se puede representar. Por ejemplo, si π se redondea a 3.14 en lugar de 3.1415926535, se introduce un error en los cálculos que lo usen.

Error de Truncamiento:

Se genera cuando se corta un número en un cierto punto sin considerar los valores restantes. Por ejemplo, truncar 3.1415926535 a 3.1415 introduce un pequeño error en comparación con el valor real.

Error de desbordamiento:

Ocurre cuando un número es demasiado grande para ser representado en la memoria del sistema. Un ejemplo común es intentar almacenar 10^{400} en un sistema con una capacidad finita de representación numérica.

Error de exactitud:

Mide qué tan cercano está un resultado numérico del valor real. Un sistema es más exacto si el error absoluto es pequeño.

Error de precisión:

Se refiere a la consistencia de los resultados numéricos al realizar cálculos repetidos. Puede ocurrir que un método tenga resultados precisos pero no exactos, si todos los valores están cercanos entre sí pero alejados del valor real.

Error absoluto:

Es la diferencia entre el valor exacto y el valor aproximado: $E_a = |X_{\text{exacto}} - X_{\text{aproximado}}|$

Error relativo:

Es el error absoluto en relación con el valor exacto: $E_r = E_a / |X_{\text{exacto}}|$

Se utiliza para evaluar la magnitud del error en proporción al valor real.

Importancia de los errores en los métodos numéricos

El análisis de estos errores es fundamental en la Ingeniería en Sistemas Computacionales, ya que muchas aplicaciones dependen de cálculos precisos. En áreas como simulaciones, inteligencia artificial y criptografía, minimizar los errores garantiza resultados confiables y mejora la eficiencia de los algoritmos.

Índice

Ejercicios.....	4
Ejercicios de Redondeo	4
Ejercicios de Truncamiento	9
Ejercicios de Desbordamiento.....	14
Ejercicios de Precisión	17
Ejercicios extra	22
Conclusiones	27
Bibliografía.....	28
Extras	29
Distribución del trabajo	29

Ejercicios

Ejercicios de Redondeo

Ejercicio 1

En un sistema financiero, los cálculos de intereses requieren precisión. Sin embargo, los sistemas computacionales redondean los valores al manejar dinero. Un banco quiere calcular el interés mensual sobre un saldo de \$10,000.257\$ usando una tasa del 3.75% anual.

Solución en Python:

```
capital = 10000.257
tasa_anual = 3.75 / 100
interes_sin_redondeo = (capital * tasa_anual) / 12

interes_redondeado = round(interes_sin_redondeo, 2)

error_absoluto = abs(interes_sin_redondeo - interes_redondeado)
error_relativo = error_absoluto / abs(interes_sin_redondeo)

print(f"Interés sin redondeo: {interes_sin_redondeo}")
print(f"Interés redondeado: {interes_redondeado}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
```

Ejecución del código:

```
PS C:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2> & 'c:\Users\diego\anaconda3\python.exe' 'c:\Users\diego\.vscode\extensions\ms-python.debugpy-2025.0.0-win32-x64\bundled\libs\debugpy\launcher' '54038' '--' 'C:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2\Ejercicios.py'
Interés sin redondeo: 31.250803125
Interés redondeado: 31.25
Error absoluto: 0.0008031250000000038
Error relativo: 2.5699339527003076e-05
PS C:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2> █
```

El error de redondeo es inevitable en sistemas financieros, pero debe minimizarse para evitar inconsistencias.

Ejercicio 2

Conversión de Temperatura

Un sistema de climatización convierte temperaturas de grados Celsius a Fahrenheit usando la fórmula:

$$F = \left(\frac{9}{5} \times C\right) + 32$$

Se tiene una temperatura de 36.123 °C. El sistema redondea el resultado a 1 decimal. Se requiere calcular el error absoluto y relativo entre el valor exacto y el redondeado.

Solución con Python:

```
C = 36.123

F_exacto = (9/5) * C + 32

F_redondeado = round(F_exacto, 1)

error_absoluto = abs(F_exacto - F_redondeado)
error_relativo = error_absoluto / abs(F_exacto)

print("Conversión de °C a °F")
print(f"Temperatura exacta: {F_exacto}")
print(f"Temperatura redondeada: {F_redondeado}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
```

Ejecución del código:

```
Conversión de °C a °F
Temperatura exacta: 97.0214
Temperatura redondeada: 97.0
Error absoluto: 0.021399999999999864
Error relativo: 0.00022056989488916738
```

El redondeo a 1 decimal introduce un pequeño error en la conversión. Aunque la diferencia es mínima, en sistemas críticos es importante conocer y controlar este tipo de error.

Ejercicio 3

Precio de un Artículo

En una tienda, el precio de un artículo se determina con alta precisión pero se muestra redondeado a dos decimales para facilitar la presentación. Si el precio exacto es \$5.6789, se redondea a \$5.68. Se debe calcular el error absoluto y relativo del redondeo.

Solución con Python:

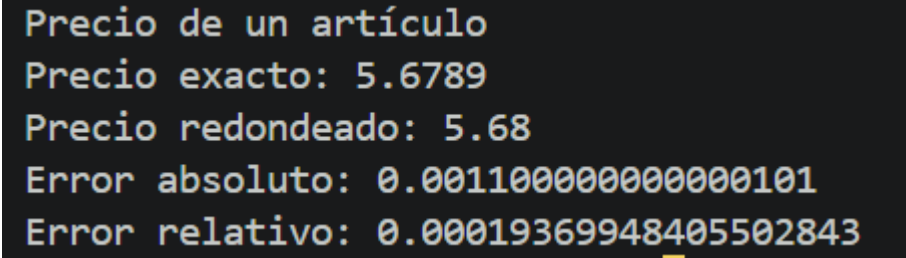
```
precio_exacto = 5.6789

precio_redondeado = round(precio_exacto, 2)

error_absoluto = abs(precio_exacto - precio_redondeado)
error_relativo = error_absoluto / abs(precio_exacto)

print("Precio de un artículo")
print(f"Precio exacto: {precio_exacto}")
print(f"Precio redondeado: {precio_redondeado}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
```

Ejecución del código:



```
Precio de un artículo
Precio exacto: 5.6789
Precio redondeado: 5.68
Error absoluto: 0.001100000000000101
Error relativo: 0.00019369948405502843
```

El redondeo a dos decimales genera un error que, aunque pequeño, puede afectar el total en operaciones a gran escala. Es importante conocer esta diferencia para ajustar procesos contables y de facturación.

Ejercicio 4

Medición de Distancia

En un sistema de navegación, una distancia medida es 123.4567 km, pero por motivos de formato se redondea a 1 decimal (es decir, 123.5 km). Se requiere calcular el error absoluto y relativo del redondeo.

Solución con Python:

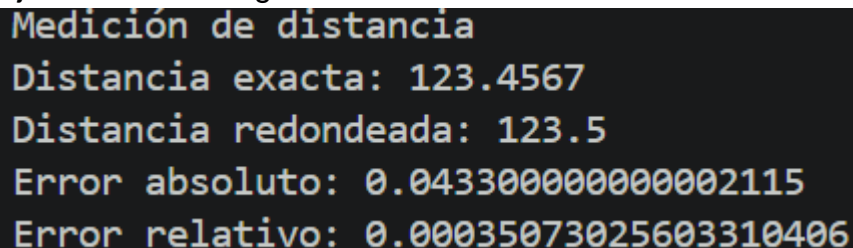
```
distancia_exacta = 123.4567

distancia_redondeada = round(distancia_exacta, 1)

error_absoluto = abs(distancia_exacta - distancia_redondeada)
error_relativo = error_absoluto / abs(distancia_exacta)

print("Medición de distancia")
print(f"Distancia exacta: {distancia_exacta}")
print(f"Distancia redondeada: {distancia_redondeada}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
```

Ejecución del código:



```
Medición de distancia
Distancia exacta: 123.4567
Distancia redondeada: 123.5
Error absoluto: 0.043300000000002115
Error relativo: 0.00035073025603310406
```

La precisión de la medición se reduce al redondear, lo que ocasiona un error pequeño pero presente. Conocer este error es útil para evaluar la exactitud de los sistemas de navegación.

Ejercicio 5

Conversión de Moneda

Un turista desea convertir euros a dólares. Tiene 50.987 euros y la tasa de cambio es de 1.12 dólares por euro. El valor exacto se obtiene multiplicando la cantidad de euros por la tasa, pero el sistema redondea el resultado a dos decimales. Se requiere calcular el error absoluto y relativo introducido por el redondeo.

Solución con Python:

```
euros = 50.987
tipo_cambio = 1.12

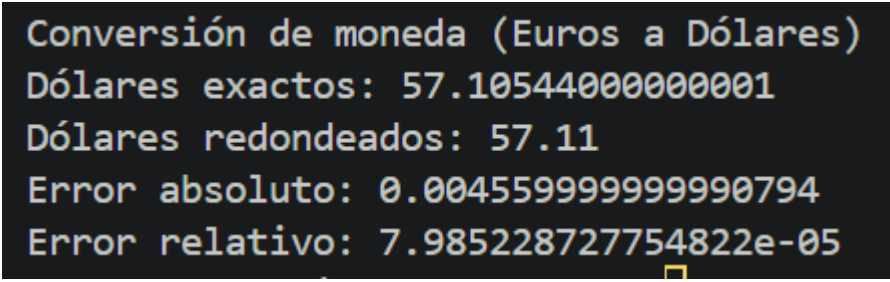
dolares_exactos = euros * tipo_cambio

dolares_redondeados = round(dolares_exactos, 2)

error_absoluto = abs(dolares_exactos - dolares_redondeados)
error_relativo = error_absoluto / abs(dolares_exactos)

print("Conversión de moneda (Euros a Dólares)")
print(f"Dólares exactos: {dolares_exactos}")
print(f"Dólares redondeados: {dolares_redondeados}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
```

Ejecución del código:



```
Conversión de moneda (Euros a Dólares)
Dólares exactos: 57.10544000000001
Dólares redondeados: 57.11
Error absoluto: 0.0045599999999990794
Error relativo: 7.985228727754822e-05
```

El redondeo a dos decimales en la conversión de moneda introduce un pequeño error. Aunque la diferencia sea mínima, es importante tener en cuenta este error en operaciones financieras donde se manejen grandes volúmenes de transacciones.

Ejercicios de Truncamiento

Ejercicio 1

Un banco desea calcular el interés mensual sobre un saldo de \$10,000.257 con una tasa de 3.75% anual. Sin embargo, por reglas del sistema financiero, el saldo debe manejarse con solo dos decimales. Se requiere:

1. Calcular el interés mensual sin truncamiento.
2. Calcular el interés mensual truncado a dos decimales.
3. Determinar el error absoluto, relativo y de precisión.

Solución en Python:

```
import math

saldo = 10000.257
tasa_anual = 3.75 / 100
tasa_mensual = tasa_anual / 12

interes_real = saldo * tasa_mensual

interes_truncado = math.trunc(interes_real * 100) / 100 # Truncamiento a dos decimales

error_absoluto = abs(interes_real - interes_truncado)
error_relativo = error_absoluto / abs(interes_real)
error_precision = abs(error_absoluto / interes_truncado) if interes_truncado != 0 else float('inf')

print("Ejercicio 1: Error por Truncamiento")
print(f"Interés real: {interes_real}")
print(f"Interés truncado: {interes_truncado}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
print(f"Error de precisión: {error_precision}")
```

Ejecución del código:

```
PS C:\Users\oscar> & C:/Users/oscar/anaconda3/python.exe "c:/Externo Mio/ITESA/ITESA 4to Semestre/Metodos Numericos/truncamiento.py"
Ejercicio 1: Error por Truncamiento
Interés real: 31.250803124999997
Interés truncado: 31.25
Error absoluto: 0.0008031249999973511
Error relativo: 2.5699339526889397e-05
Error de precisión: 2.569999999915234e-05
```

Ejercicio 2:

En ingeniería civil, el cálculo de áreas es esencial para determinar la cantidad de materiales necesarios. Un ingeniero calcula el área de un terreno circular con un radio de 7.823 metros, pero por restricciones de software, solo se permiten dos decimales. Se requiere:

1. Calcular el área real del círculo.
2. Calcular el área truncada a dos decimales.
3. Determinar el error absoluto, relativo y de precisión.

Solución en Python:

```
import math

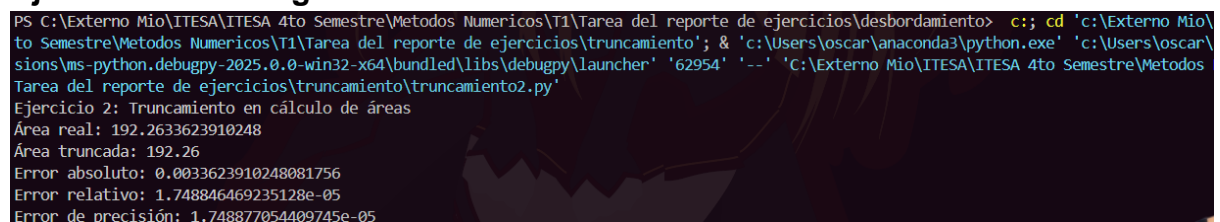
radio = 7.823
area_real = math.pi * (radio ** 2)

area_truncada = math.trunc(area_real * 100) / 100

error_absoluto = abs(area_real - area_truncada)
error_relativo = error_absoluto / abs(area_real)
error_precision = abs(error_absoluto / area_truncada) if area_truncada != 0
else float('inf')

print("Ejercicio 2: Truncamiento en cálculo de áreas")
print(f"Área real: {area_real}")
print(f"Área truncada: {area_truncada}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
print(f"Error de precisión: {error_precision}")
```

Ejecución del código:



```
PS C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\desbordamiento> c.; cd 'c:\Externo Mio\to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento'; & 'c:\Users\oscar\anaconda3\python.exe' 'c:\Users\oscar\sions\ms-python.debugpy-2025.0.0-win32-x64\bundled\libs\debugpy\launcher' '62954' '--' 'C:\Externo Mio\ITESA\ITESA 4to Semestre\MetodosTarea del reporte de ejercicios\truncamiento\truncamiento2.py'
Ejercicio 2: Truncamiento en cálculo de áreas
Área real: 192.2633623910248
Área truncada: 192.26
Error absoluto: 0.0033623910248081756
Error relativo: 1.748846469235128e-05
Error de precisión: 1.748877054409745e-05
```

Ejercicio 3:

Un meteorólogo convierte la temperatura de 37.857°C a Fahrenheit, pero el informe solo permite dos decimales. Se requiere:

1. Calcular la temperatura exacta en Fahrenheit.
2. Calcular la temperatura truncada a dos decimales.
3. Determinar el error absoluto, relativo y de precisión.

Solución en Python:

```
import math

celsius = 37.857
fahrenheit_real = (celsius * 9/5) + 32

fahrenheit_truncado = math.trunc(fahrenheit_real * 100) / 100

error_absoluto = abs(fahrenheit_real - fahrenheit_truncado)
error_relativo = error_absoluto / abs(fahrenheit_real)
error_precision = abs(error_absoluto / fahrenheit_truncado) if
fahrenheit_truncado != 0 else float('inf')

print("Ejercicio 3: Truncamiento en conversión de temperatura")
print(f"Temperatura real: {fahrenheit_real}")
print(f"Temperatura truncada: {fahrenheit_truncado}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
print(f"Error de precisión: {error_precision}")
```

Ejecución del código:

```
PS C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento> c:: cd 'c:\Externo Mio\
Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento'; & 'c:\Users\oscar\anaconda3\python.exe' 'c:\Users\oscar\
ons\ms-python-debugpy-2025.0.0-win32-x64\bundle\libs\debugpy\launcher' '63029' '---' 'C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos N
rea del reporte de ejercicios\truncamiento\truncamiento3.py'
Ejercicio 3: Truncamiento en conversión de temperatura
Temperatura real: 100.14259999999999
Temperatura truncada: 100.14
Error absoluto: 0.002599999999986835
Error relativo: 2.5962976794958744e-05
Error de precisión: 2.5963650888624278e-05
```

Ejercicio 4:

Un físico mide la velocidad de un objeto en 12.987 m/s, pero el reporte solo permite dos decimales. Se requiere:

1. Calcular la velocidad sin truncamiento.
2. Calcular la velocidad truncada a dos decimales.
3. Determinar los errores absoluto, relativo y de precisión.

Solución en Python:

```
import math

velocidad_real = 12.987

velocidad_truncada = math.trunc(velocidad_real * 100) / 100

error_absoluto = abs(velocidad_real - velocidad_truncada)
error_relativo = error_absoluto / abs(velocidad_real)
error_precision = abs(error_absoluto / velocidad_truncada) if
velocidad_truncada != 0 else float('inf')

print("Ejercicio 4: Truncamiento en velocidad")
print(f"Velocidad real: {velocidad_real}")
print(f"Velocidad truncada: {velocidad_truncada}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
print(f"Error de precisión: {error_precision}")
```

Ejecución del código:

```
PS C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento> c:; cd 'c:\Externo Mio\
Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento'; & 'c:\Users\oscar\anaconda3\python.exe' 'c:\Users\oscar\
ons\ms-python.debugpy-2025.0.0-win32-x64\bundle\libs\debugpy\launcher' '63085' '--' 'c:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos N
rea del reporte de ejercicios\truncamiento\truncamiento4.py'
Ejercicio 4: Truncamiento en velocidad
Velocidad real: 12.987
Velocidad truncada: 12.98
Error absoluto: 0.0069999999999999673
Error relativo: 0.0005390005390005138
Error de precisión: 0.0005392912172572938
```

Ejercicio 5:

Un químico calcula la densidad de un líquido como 0.99876 g/cm^3 , pero su informe solo permite tres decimales. Se requiere:

1. Calcular la densidad real.
2. Calcular la densidad truncada a tres decimales.
3. Determinar el error absoluto, relativo y de precisión.

Solución en Python:

```
import math

densidad_real = 0.99876

densidad_truncada = math.trunc(densidad_real * 1000) / 1000

error_absoluto = abs(densidad_real - densidad_truncada)
error_relativo = error_absoluto / abs(densidad_real)
error_precision = abs(error_absoluto / densidad_truncada) if
densidad_truncada != 0 else float('inf')

print("Ejercicio 5: Truncamiento en densidad")
print(f"Densidad real: {densidad_real}")
print(f"Densidad truncada: {densidad_truncada}")
print(f"Error absoluto: {error_absoluto}")
print(f"Error relativo: {error_relativo}")
print(f"Error de precisión: {error_precision}")
```

Ejecución del código:

```
PS C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento> c:; cd 'c:\Externo Mio\
Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\truncamiento'; & 'c:\Users\oscar\anaconda3\python.exe' 'c:\Users\oscar\
ons\ms-python-debugpy-2025.0.0-win32-x64\bundled\libs\debugpy\launcher' '63138' '---' 'c:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos
rea del reporte de ejercicios\truncamiento\truncamiento5.py'
Ejercicio 5: Truncamiento en densidad
Densidad real: 0.99876
Densidad truncada: 0.998
Error absoluto: 0.0007599999999999829
Error relativo: 0.0007609435700268161
Error de precisión: 0.0007615230460921673
```


Ejercicio 2:

Un matemático calcula la suma de una progresión geométrica con $n = 200$ y razón $r = 2$, pero el sistema puede sufrir desbordamiento.

Solución en Python:

```
def suma_geometrica(n, r):
    if n == 0:
        return 1
    return r * suma_geometrica(n - 1, r)

try:
    suma = suma_geometrica(200, 2)
except RecursionError:
    suma = "Error por desbordamiento"

print("Ejercicio 2: Desbordamiento en progresión geométrica")
print(f"Suma de la serie: {suma}")
```

Ejecución del código:

PS C:\Users\oscar> & c:\Users\oscar\anaconda3\python.exe "c:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de eje
s/desbordamiento/desbordamiento1.py"
Ejercicio 2: Desbordamiento en progresión geométrica
Suma de la serie: 1606938044258990275541962092341162602522202993782792835301376

Ejercicio 3:

Un físico simula la energía liberada en una reacción nuclear con la ecuación $E = m \cdot c^2$, pero valores grandes de masa pueden causar desbordamiento.

Solución en Python:

```
m = 10**30
c = 3 * 10**8

try:
    energia = m * c**2
except OverflowError:
    energia = "Error por desbordamiento"

print("Ejercicio 3: Desbordamiento en energía")
print(f"Energía liberada: {energia}")
```

Ejecución del código:

[illegible]

Un biólogo usa la secuencia de Fibonacci para modelar el crecimiento de una población de bacterias. Intenta calcular el término $F(1000)$ de la serie, pero el crecimiento exponencial de los valores puede causar un desbordamiento.

```
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    return fibonacci(n - 1) + fibonacci(n - 2)

try:
    fib_1000 = fibonacci(1000)
except RecursionError:
    fib_1000 = "Error por desbordamiento"

print("Ejercicio 4: Desbordamiento en la serie de Fibonacci")
print(f"Fibonacci(1000): {fib_1000}")
```

```
PS C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\desbordamiento> c;; cd 'c:\Externo Mio\Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\desbordamiento'; & 'c:\Users\oscar\anaconda3\python.exe' 'c:\Users\oscar\anaconda3\python\python-debugpy-2025.0.0-win32-x64\bundle\libs\debugpy\launcher' '62763' '--' 'c:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\Tarea del reporte de ejercicios\desbordamiento\desbordamiento3.py'
```

Una empresa almacena transacciones financieras en una base de datos y necesita calcular la sumatoria total de 10 millones de valores. Si la suma excede el límite del sistema, se producirá un desbordamiento numérico.

```
import sys

transacciones = [sys.maxsize] * 10_000_000

try:
    total = sum(transacciones)
except OverflowError:
    total = "Error por desbordamiento"

print("Ejercicio 5: Desbordamiento en sumatoria financiera")
print(f"Total de transacciones: {total}")
```

```
PS C:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\tarea del reporte de ejercicios\desbordamiento> cd "c:\Externo Mio\ITESA\ITESA 4to Semestre\Metodos Numericos\T1\tarea del reporte de ejercicios\desbordamiento"; & "c:\Users\oscar\anaconda3\python.exe" "c:\Users\oscar\AppData\Local\Programs\Python\Python38-64\Scripts\python.py" -m sys.path.append("C:/Externo/Mio/ITESA/ITESA 4to Semestre/Metodos Numericos/T1/tarea del reporte de ejercicios/desbordamiento")
```

```
Ejercicio 5: Desbordamiento en sumatoria financiera  
Total de transacciones: 92233720368547758070000000
```


Ejercicios de Precisión

Ejercicio 1

Un ingeniero está diseñando una pieza mecánica que debe ajustarse con una tolerancia de 0.0001 mm. Sin embargo, al usar diferentes precisiones en los cálculos, obtiene distintos valores para el área de un cilindro.

El área lateral de un cilindro se calcula como: $A = 2\pi rh$

Donde:

$r = 5.6732$ cm es el radio del cilindro.

$h = 12.8913$ cm es la altura del cilindro.

π es aproximadamente 3.1415926535.

Solución en Python:

```
import math

r = 5.6732
h = 12.8913
pi_exacto = math.pi # Usa la mayor precisión disponible en Python

area_exacta = 2 * pi_exacto * r * h

area_3_decimales = round(area_exacta, 3)
area_5_decimales = round(area_exacta, 5)
area_10_decimales = round(area_exacta, 10)

def calcular_errores(area_aprox):
    error_absoluto = abs(area_exacta - area_aprox)
    error_relativo = error_absoluto / abs(area_exacta)
    return error_absoluto, error_relativo

error_abs_3, error_rel_3 = calcular_errores(area_3_decimales)
error_abs_5, error_rel_5 = calcular_errores(area_5_decimales)
error_abs_10, error_rel_10 = calcular_errores(area_10_decimales)

print(f"Área exacta: {area_exacta}")
print(f"Truncada a 3 decimales: {area_3_decimales}, Error absoluto: {error_abs_3}, Error relativo: {error_rel_3}")
print(f"Truncada a 5 decimales: {area_5_decimales}, Error absoluto: {error_abs_5}, Error relativo: {error_rel_5}")
print(f"Truncada a 10 decimales: {area_10_decimales}, Error absoluto: {error_abs_10}, Error relativo: {error_rel_10}")
```

Ejecución del código:

```
PS C:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2> c;; cd 'c:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2'; & 'c:\Users\diego\anaconda3\python.exe' 'c:\Users\diego\.vscode\extensions\ms-python.debugpy-2025.0.0-win32-x64\bundled\libs\debugpy\launcher' '54265' '--' 'C:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2\Ejercicios.py'
Área exacta: 459.52027464062
Truncada a 3 decimales: 459.52, Error absoluto: 0.0002746406200344609, Error relativo: 5.976681230207978e-07
Truncada a 5 decimales: 459.52027, Error absoluto: 4.640620034024323e-06, Error relativo: 1.009883630848202e-08
Truncada a 10 decimales: 459.5202746406, Error absoluto: 2.000888343900442e-11, Error relativo: 4.3542982852395134e-14
PS C:\Users\diego\OneDrive\Documentos\ITESA\CUARTO_SEMESTRE\4F21_Metodos_Numericos\Tema_1\Evidencia2> █
```

El error de precisión se reduce al aumentar el número de decimales, lo cual es crítico en cálculos de manufactura y diseño mecánico.

Ejercicio 2

Área de un Círculo

Se desea calcular el área de un círculo con radio 2.3456 cm usando la fórmula:

$$A = \pi \times r^2$$

Se calculará el área exacta (usando la mayor precisión de Python) y se mostrará el resultado aproximado redondeado a 2, 4 y 6 decimales. Se calculará el error absoluto y relativo para cada aproximación.

Solución en Python:

```
import math

r = 2.3456

area_exacta = math.pi * r**2

area_2_dec = round(area_exacta, 2)
area_4_dec = round(area_exacta, 4)
area_6_dec = round(area_exacta, 6)

def calcular_errores(valor_exacto, valor_aprox):
    error_absoluto = abs(valor_exacto - valor_aprox)
    error_relativo = error_absoluto / abs(valor_exacto)
    return error_absoluto, error_relativo

error_2, rel_2 = calcular_errores(area_exacta, area_2_dec)
error_4, rel_4 = calcular_errores(area_exacta, area_4_dec)
error_6, rel_6 = calcular_errores(area_exacta, area_6_dec)

print("Área de un círculo")
print(f"Área exacta: {area_exacta}")
print(f"Redondeado a 2 decimales: {area_2_dec} | Error absoluto: {error_2} | Error relativo: {rel_2}")
print(f"Redondeado a 4 decimales: {area_4_dec} | Error absoluto: {error_4} | Error relativo: {rel_4}")
print(f"Redondeado a 6 decimales: {area_6_dec} | Error absoluto: {error_6} | Error relativo: {rel_6}")
```

Ejecución del código:

```
Área de un círculo
Área exacta: 17.284538114607173
Redondeado a 2 decimales: 17.28 | Error absoluto: 0.004538114607171906 | Error relativo: 0.00026255342069781676
Redondeado a 4 decimales: 17.2845 | Error absoluto: 3.8114607171735315e-05 | Error relativo: 2.2051273177803136e-06
Redondeado a 6 decimales: 17.284538 | Error absoluto: 1.1460717175282298e-07 | Error relativo: 6.630618127768679e-09
```

La precisión en el cálculo del área mejora al aumentar la cantidad de decimales en la representación. Esto es crucial en aplicaciones de ingeniería donde se requiere alta exactitud en el diseño.

Ejercicio 3

Valor de la Función Exponencial

Calcular el valor de la función exponencial e^x para $x=1.2345$ usando `math.exp`. Se obtendrá el valor exacto y se presentarán las aproximaciones redondeadas a 2, 4 y 6 decimales, calculando el error absoluto y relativo en cada caso.

Solución en Python:

```
import math

x = 1.2345

valor_exacto = math.exp(x)

valor_2_dec = round(valor_exacto, 2)
valor_4_dec = round(valor_exacto, 4)
valor_6_dec = round(valor_exacto, 6)

def calcular_errores(valor_exacto, valor_aprox):
    error_absoluto = abs(valor_exacto - valor_aprox)
    error_relativo = error_absoluto / abs(valor_exacto)
    return error_absoluto, error_relativo

error_2, rel_2 = calcular_errores(valor_exacto, valor_2_dec)
error_4, rel_4 = calcular_errores(valor_exacto, valor_4_dec)
error_6, rel_6 = calcular_errores(valor_exacto, valor_6_dec)

print("Valor de e^x para x=1.2345")
print(f"Valor exacto: {valor_exacto}")
print(f"Redondeado a 2 decimales: {valor_2_dec} | Error absoluto: {error_2} | Error relativo: {rel_2}")
print(f"Redondeado a 4 decimales: {valor_4_dec} | Error absoluto: {error_4} | Error relativo: {rel_4}")
print(f"Redondeado a 6 decimales: {valor_6_dec} | Error absoluto: {error_6} | Error relativo: {rel_6}")
```

Ejecución del código:

```
Valor de e^x para x=1.2345
Valor exacto: 3.436659761170463
Redondeado a 2 decimales: 3.44 | Error absoluto: 0.0033402388295371566 | Error relativo: 0.000971943416475867
Redondeado a 4 decimales: 3.4367 | Error absoluto: 4.023882953729796e-05 | Error relativo: 1.1708703314753904e-05
Redondeado a 6 decimales: 3.43666 | Error absoluto: 2.388295370359117e-07 | Error relativo: 6.949467030002724e-08
```

La representación numérica del valor exponencial mejora significativamente con mayor precisión decimal. Este ejercicio destaca la importancia de ajustar la precisión en función de la sensibilidad requerida en el análisis.

Ejercicio 4

Promedio de Números

Se tienen tres números: 12.345, 67.890 y 23.456. Se desea calcular el promedio exacto y luego presentar este promedio redondeado a 2, 4 y 6 decimales. Se calcularán el error absoluto y relativo de cada aproximación.

Solución en Python:

```
num1 = 12.345
num2 = 67.890
num3 = 23.456

promedio_exacto = (num1 + num2 + num3) / 3

prom_2_dec = round(promedio_exacto, 2)
prom_4_dec = round(promedio_exacto, 4)
prom_6_dec = round(promedio_exacto, 6)

def calcular_errores(valor_exacto, valor_aprox):
    error_absoluto = abs(valor_exacto - valor_aprox)
    error_relativo = error_absoluto / abs(valor_exacto)
    return error_absoluto, error_relativo

error_2, rel_2 = calcular_errores(promedio_exacto, prom_2_dec)
error_4, rel_4 = calcular_errores(promedio_exacto, prom_4_dec)
error_6, rel_6 = calcular_errores(promedio_exacto, prom_6_dec)

print("Cálculo del promedio de tres números")
print(f"Promedio exacto: {promedio_exacto}")
print(f"Redondeado a 2 decimales: {prom_2_dec} | Error absoluto: {error_2} | Error relativo: {rel_2}")
print(f"Redondeado a 4 decimales: {prom_4_dec} | Error absoluto: {error_4} | Error relativo: {rel_4}")
print(f"Redondeado a 6 decimales: {prom_6_dec} | Error absoluto: {error_6} | Error relativo: {rel_6}")
```

Ejecución del código:

```
Cálculo del promedio de tres números
Promedio exacto: 34.56366666666667
Redondeado a 2 decimales: 34.56 | Error absoluto: 0.003666666666667595 | Error relativo: 0.0001060844239133848
Redondeado a 4 decimales: 34.5637 | Error absoluto: 3.333333332733446e-05 | Error relativo: 9.644038535842394e-07
Redondeado a 6 decimales: 34.563667 | Error absoluto: 3.3333333249174757e-07 | Error relativo: 9.644038513229138e-09
```

El ejercicio demuestra que la representación del promedio varía sutilmente según la precisión utilizada. Un mayor número de decimales reduce el error, lo cual es fundamental cuando se requiere alta exactitud en los análisis estadísticos.

Ejercicio 5

Cálculo de la Raíz Cuadrada de 2

Se desea calcular la raíz cuadrada de 2 utilizando la función `math.sqrt` de Python para obtener el valor con la mayor precisión posible. Posteriormente, se presentarán aproximaciones redondeadas a 2, 4 y 6 decimales, calculando el error absoluto y relativo en cada caso.

Solución en Python:

```
import math

valor_exacto = math.sqrt(2)

valor_2_dec = round(valor_exacto, 2)
valor_4_dec = round(valor_exacto, 4)
valor_6_dec = round(valor_exacto, 6)

def calcular_errores(valor_exacto, valor_aprox):
    error_absoluto = abs(valor_exacto - valor_aprox)
    error_relativo = error_absoluto / abs(valor_exacto)
    return error_absoluto, error_relativo

error_2, rel_2 = calcular_errores(valor_exacto, valor_2_dec)
error_4, rel_4 = calcular_errores(valor_exacto, valor_4_dec)
error_6, rel_6 = calcular_errores(valor_exacto, valor_6_dec)

print("Cálculo de la raíz cuadrada de 2")
print(f"Valor exacto: {valor_exacto}")
print(f"Redondeado a 2 decimales: {valor_2_dec} | Error absoluto: {error_2} | Error relativo: {rel_2}")
print(f"Redondeado a 4 decimales: {valor_4_dec} | Error absoluto: {error_4} | Error relativo: {rel_4}")
print(f"Redondeado a 6 decimales: {valor_6_dec} | Error absoluto: {error_6} | Error relativo: {rel_6}")
```

Ejecución del código:

```
Cálculo de la raíz cuadrada de 2
Valor exacto: 1.4142135623730951
Redondeado a 2 decimales: 1.41 | Error absoluto: 0.004213562373095225 | Error relativo: 0.0029794385269681155
Redondeado a 4 decimales: 1.4142 | Error absoluto: 1.3562373095243885e-05 | Error relativo: 9.590045984628936e-06
Redondeado a 6 decimales: 1.414214 | Error absoluto: 4.376269049366499e-07 | Error relativo: 3.094489521103857e-07
```

El ejercicio demuestra que al aumentar la cantidad de decimales en la representación del valor de la raíz cuadrada de 2 se reduce significativamente el error absoluto y relativo. Esto resalta la importancia de ajustar la precisión según los requerimientos de exactitud en aplicaciones científicas y de ingeniería.

Ejercicios extra

Ejercicio 1

Una empresa ofrece préstamos con una tasa anual del 5.25%. Un cliente solicita un préstamo de \$15,250.789 y desea saber cuánto pagará de intereses mensuales antes y después del redondeo a dos decimales.

Solución en Java:

```
public class Main
{
    public static void main(String[] args) {

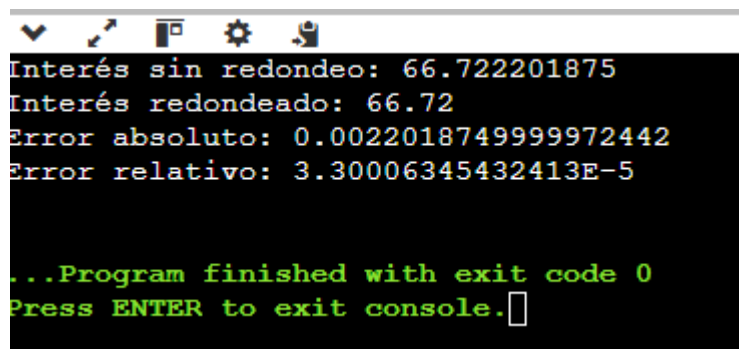
        double capital = 15250.789;
        double tasaAnual = 5.25 / 100;

        double interesSinRedondeo = (capital * tasaAnual) / 12;
        double interesRedondeado = Math.round(interесSinRedondeo
* 100.0) / 100.0;

        double errorAbsoluto = Math.abs(interесSinRedondeo -
interесRedondeado);
        double errorRelativo = errorAbsoluto /
Math.abs(interесSinRedondeo);

        System.out.println("Interés sin redondeo: " +
interесSinRedondeo);
        System.out.println("Interés redondeado: " +
interесRedondeado);
        System.out.println("Error absoluto: " + errorAbsoluto);
        System.out.println("Error relativo: " + errorRelativo);
    }
}
```

Ejecución del código:



```
Interés sin redondeo: 66.722201875
Interés redondeado: 66.72
Error absoluto: 0.0022018749999972442
Error relativo: 3.30006345432413E-5

...Program finished with exit code 0
Press ENTER to exit console.
```

Ejercicio 2

Una tienda ofrece un descuento del 12.5% en una compra de \$259.975. Sin embargo, el sistema solo permite mostrar el descuento con dos decimales.

Objetivo:

- ❖ Calcular el descuento sin redondear.
- ❖ Redondearlo a dos decimales.
- ❖ Calcular el error absoluto y relativo.

Solución en Java:

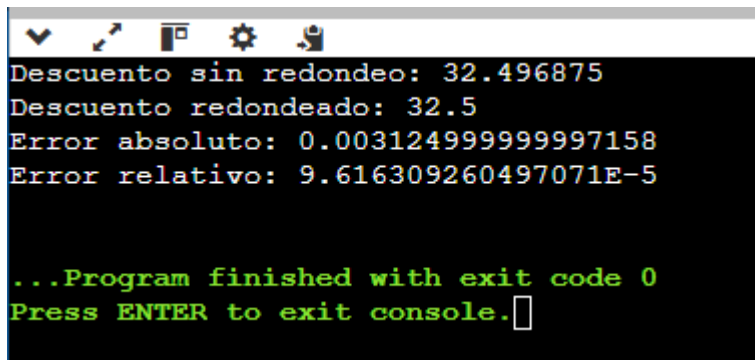
```
import java.util.Scanner;
public class Main {
    public static void main(String[] args) {
        double precio = 259.975;
        double descuentoPorcentaje = 12.5 / 100;
        double descuentoSinRedondeo = precio * descuentoPorcentaje;

        double descuentoRedondeado = Math.round(descuentoSinRedondeo *
100.0) / 100.0;

        double errorAbsoluto = Math.abs(descuentoSinRedondeo -
descuentoRedondeado);
        double errorRelativo = errorAbsoluto /
Math.abs(descuentoSinRedondeo);

        System.out.println("Descuento sin redondeo: " +
descuentoSinRedondeo);
        System.out.println("Descuento redondeado: " + descuentoRedondeado);
        System.out.println("Error absoluto: " + errorAbsoluto);
        System.out.println("Error relativo: " + errorRelativo);
    }
}
```

Ejecución del código:



```
Descuento sin redondeo: 32.496875
Descuento redondeado: 32.5
Error absoluto: 0.003124999999997158
Error relativo: 9.616309260497071E-5

...Program finished with exit code 0
Press ENTER to exit console.
```

Ejercicio 3

Un turista quiere cambiar \$150.875 dólares a euros con una tasa de cambio de 1 dólar = 0.9234 euros. El banco solo permite mostrar la conversión con dos decimales.

Objetivo:

- ❖ Calcular la conversión sin redondeo.
- ❖ Redondearla a dos decimales.
- ❖ Calcular el error absoluto y relativo.

Solución en Java:

```
import java.util.Scanner;

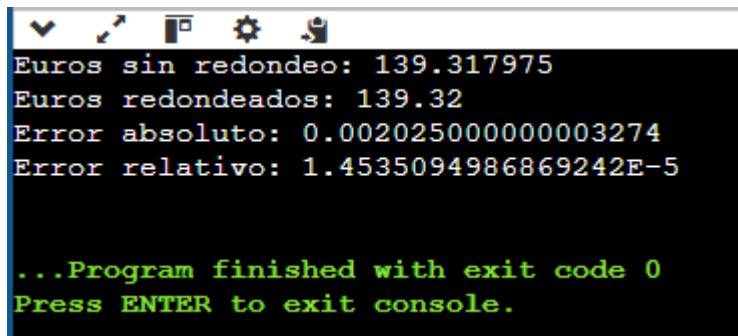
public class Main {
    public static void main(String[] args) {
        double dolares = 150.875;
        double tasaCambio = 0.9234;
        double eurosSinRedondeo = dolares * tasaCambio;

        double eurosRedondeados = Math.round(eurosSinRedondeo * 100.0) /
100.0;

        double errorAbsoluto = Math.abs(eurosSinRedondeo -
eurosRedondeados);
        double errorRelativo = errorAbsoluto / Math.abs(eurosSinRedondeo);

        System.out.println("Euros sin redondeo: " + eurosSinRedondeo);
        System.out.println("Euros redondeados: " + eurosRedondeados);
        System.out.println("Error absoluto: " + errorAbsoluto);
        System.out.println("Error relativo: " + errorRelativo);
    }
}
```


Ejecución del código:



```
▼ ↗ 📄 ⚙️ 👤
Euros sin redondeo: 139.317975
Euros redondeados: 139.32
Error absoluto: 0.002025000000003274
Error relativo: 1.4535094986869242E-5

...Program finished with exit code 0
Press ENTER to exit console.
```

Ejercicio 3

Una empresa necesita calcular el impuesto mensual sobre una venta de \$12,589.8765 con una tasa de 16%. Sin embargo, el sistema de facturación solo acepta valores truncados a dos decimales en los cálculos.

La tarea es calcular:

1. El impuesto sin truncar
2. El impuesto truncado a dos decimales
3. El error absoluto y relativo debido al truncamiento

Solución en Excel:

Concepto	Valor
Venta	12589.8765
Tasa de impuesto	0.16
Impuesto sin truncar	2014.38024
Impuesto truncado	2014.38
Error absoluto	0.00024
Error relativo	0.0000001191

Ejercicio 4

Una empresa paga un salario mensual de \$12,567.89 y quiere calcular el salario diario considerando 30 días. Sin embargo, el resultado debe truncarse a dos decimales. Además, se calcularán el error absoluto y relativo respecto al valor real.

Solución en Excel:

A	B	C
Salario mensual	12567.89	
Número de días trabajados	30	
Salario diario sin truncar	418.9296667	
Salario diario truncado	418.92	
Error absoluto	0.009666667	
Error relativo	0.0000230747	

Conclusiones

Oscar Aaron Delgadillo Fernandez:

Los errores de truncamiento y desbordamiento afectan la precisión de los cálculos numéricos en sistemas computacionales. El truncamiento reduce la exactitud al limitar los decimales, siendo un gran problema en áreas como las finanzas, mientras que el desbordamiento ocurre al exceder la capacidad de almacenamiento, como en el cálculo de factoriales grandes. Estos problemas nos hacen ver la importancia de seleccionar métodos adecuados para estar seguros de la precisión y evitar fallos en aplicaciones críticas.

Diego Alonso Coronel Vargas

El manejo de los errores numéricos es muy importante en cualquier proceso computacional que requiera precisión. Los errores pueden surgir por redondeo, truncamiento, desbordamiento, exactitud o precisión, afectando la confiabilidad de los cálculos. En particular, los ejercicios realizados demostraron cómo el redondeo impacta en cálculos financieros, modificando ligeramente los resultados, mientras que la precisión afecta mediciones en ingeniería, dependiendo del número de decimales utilizados. Comprender estos errores permite minimizar su impacto y garantizar mayor exactitud en aplicaciones prácticas.

Brandon García Ordaz.

En los ejercicios realizados, se pudo observar cómo los errores numéricos afectan los cálculos financieros y contables. En el caso del redondeo, vimos que al calcular intereses bancarios, los valores finales pueden variar dependiendo de cuántos decimales se conserven. Por otro lado, en el truncamiento, aplicado al cálculo de impuestos en Excel, se evidenció cómo la eliminación de decimales sin redondeo puede generar pequeñas diferencias en los resultados. Estos errores, aunque parezcan mínimos, pueden impactar significativamente en procesos financieros o de facturación. Por ello, comprender su efecto y aprender a calcular el error absoluto y relativo permite evaluar su magnitud y tomar mejores decisiones para garantizar mayor precisión en los cálculos.

Bibliografía

Chapra, S. (2007). *Metodos Numericos Para Ingenieros*. McGraw-Hill
Interamericana.

Extras

Distribución del trabajo

Gráfico Gantt

	Domingo: 5 pm	Domingo: 6 pm	Domingo: 7 pm	Domingo: 8 pm	Domingo: 9 pm	Domingo: 10 pm
Ejercicios de Redondeo	Diego					
Ejercicios de Truncamiento	Oscar					
Ejerciso de Desbordamiento	Oscar					
Ejercicios de Precisión	Diego					
Ejercicio extra 1			Brandon			
Ejercicio extra 2			Brandon			

Tabla Scrum

Diego Alonso Coronel Vargas	Brandon García Ordaz	Oscar Aaron Delgadillo Fernández
<ul style="list-style-type: none"> - Ejercicio de redondeo - Ejercicio de precisión 	<ul style="list-style-type: none"> - Ejercicio extra de redondeo - Ejercicio extra de truncación 	<ul style="list-style-type: none"> - Ejercicio de truncamiento. - Ejercicio de desbordamiento.

Evidencias de trabajo colaborativo

